Ion Măndoiu
Alexander Zelikovsky (Eds.)

LNBI 4463

Bioinformatics Research and Applications

Third International Symposium, ISBRA 2007
Atlanta, GA, USA, May 2007
Proceedings

Springer

Ion Măndoiu   Alexander Zelikovsky (Eds.)

Bioinformatics Research and Applications

Third International Symposium, ISBRA 2007
Atlanta, GA, USA, May 7-10, 2007
Proceedings

Springer

Volume Editors

Ion Măndoiu
University of Connecticut
Computer Science and Engineering Department
Storrs, CT 06269-2155, USA
E-mail: ion@engr.uconn.edu

Alexander Zelikovsky
Georgia State University
Department of Computer Science
Atlanta, GA 30303-3086, USA
E-mail: alexz@cs.gsu.edu

Library of Congress Control Number: 2007925296

CR Subject Classification (1998): J.3, H.2.8, F.1, F.2.2, G.3

LNCS Sublibrary: SL 8 – Bioinformatics

ISSN 0302-9743
ISBN-10 3-540-72030-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-72030-0 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12050161 06/3180 5 4 3 2 1 0

Preface

The 2007 International Symposium on Bioinformatics Research and Applications (ISBRA 2007), was held on May 7–10, 2007 at Georgia State University in Atlanta, Georgia. The ISBRA symposium provides a forum for the exchange of ideas and results among researchers, developers, and practitioners working on all aspects of bioinformatics and computational biology and their applications. ISBRA is the successor of the International Workshop on Bioinformatics Research and Applications (IWBRA), held May 22–25, 2005 in Atlanta, GA and May 28–31, 2006 in Reading, UK, in conjunction with the International Conference on Computational Science.

This year, 146 papers were submitted in response to the call for papers. Following a rigorous review process, the Program Committee selected 55 papers for publication in the proceedings and oral presentations at the symposium. The topics of selected papers covered a wide range of topics, including clustering and classification, gene expression analysis, gene networks, genome analysis, motif finding, pathways, protein structure prediction, protein domain interactions, phylogenetics, and software tools.

In addition to contributed talks, the ISBRA 2007 technical program included several tutorials and poster sessions, and features invited keynote talks by three distinguished speakers. Ming Li from University of Waterloo spoke on modern homology search, Laura L. Elnitski from the National Human Genome Research Institute spoke on bidirectional promoters in the human genome, and Mark Borodovsky from Georgia Institute of Technology spoke on *ab initio* gene finding.

We would like to thank all authors for submitting papers and presenting their work at the symposium. We would also like to thank the Program Committee members and external reviewers for volunteering their time and expertise to review and select symposium papers. We would like to extend special thanks to the Organizing, Publications, Finance, Publicity, and Posters Chairs, all of whom are listed on the following page, for their tremendous efforts in making ISBRA 2007 a great success. Last but not least, we would like to thank the General Chairs, Dan Gusfield and Yi Pan, for their leadership and guidance.

We hope you will find the technical program interesting and thought provoking. Enjoy!

May 2007

Ion Măndoiu
Alexander Zelikovsky

Conference Organization

General Chairs

Dan Gusfield, University of California, Davis, USA
Yi Pan, Georgia State University, USA

Program Chairs

Ion Măndoiu, University of Connecticut, USA
Alexander Zelikovsky, Georgia State University, USA

Organizing Chairs

Robert Harrison, Georgia State University, USA
Yanqing Zhang, Georgia State University, USA

Publications Chair

Raj Sunderraman, Georgia State University, USA

Finance Chairs

Anu Bourgeois, Georgia State University, USA
Akshaye Dhawan, Georgia State University, USA

Publicity Chairs

Kim King, Georgia State University, USA
Yingshu Li, Georgia State University, USA

Poster Chairs

Gulsah Altun, Georgia State University, USA
Dumitru Brinza, Georgia State University, USA

Program Committee

Gabriela Alexe
IBM Research, USA

Yonatan Aumann
Bar Ilan University, Israel

Danny Barash
Ben-Gurion University, Israel

Anne Bergeron
Université du Québec à Montréal, Canada

Piotr Berman
Penn State, USA

Paola Bonizzoni
Università degli Studi di Milano-Bicocca, Italy

Mark Borodovsky
Georgia Institute of Technology, USA

Anu Bourgeois
Georgia State University, USA

Daniel Brown
University of Waterloo, Canada

Liming Cai
University of Georgia, USA

Jake Yue Chen
IUPUI, USA

Luonan Chen
Osaka Sangyo University, Japan

Yixin Chen
Washington University, USA

Peter Damaschke
Chalmers University, Sweden

Bhaskar Dasgupta
University of Illinois at Chicago, USA

Sorin Draghici
Wayne State University, USA

Jun-Tao Guo
University of Georgia, USA

Robert Harrison
Georgia State University, USA

Jieyue He
Southeast University, China

Xiaohua Hu
Drexel University, USA

Hae-Jin Hu
Georgia State University, USA

Chun-Hsi Huang
University of Connecticut, USA

Ming-Yang Kao
Northwestern University, USA

Marek Karpinski
University of Bonn, Germany

John Kececioglu
University of Arizona, USA

Ed Keedwell
University of Exeter, UK

Guojun Li
University of Georgia, USA

Yiming Li
National Chiao Tung University, Taiwan

Yingshu Li
Georgia State University, USA

Jianzhong Li
Harbin Institute of Technology, China

Yixue Li
Shanghai Center for Bioinformation Technology, China

Guohui Lin
University of Alberta, Canada

Xiaohui Liu
Brunel University, UK

Shiyong Lu
Wayne State University, USA

Jingchu Luo
Peking University, China

Osamu Maruyama
Kyushu University, Japan

Kayvan Najarian
UNC at Charlotte, USA

Giri Narasimhan
Florida International University, USA

Craig Nelson
University of Connecticut, USA

Laxmi Parida
IBM T.J. Watson Research Center, USA

Mihai Pop
University of Maryland, USA

Alex Pothen
Old Dominion University, USA

Teresa Przytycka
NCBI, USA

Sven Rahmann
Universitắt Bielefeld, Germany

Sanguthevar Rajasekaran
University of Connecticut, USA

David Sankoff
University of Ottawa, Canada

Russell Schwartz
Carnegie Mellon University, USA

Hagit Shatkay
Queen's University, Canada

Jens Stoye
Universitắt Bielefeld, Germany

Raj Sunderraman
Georgia State University, USA

Sing-Hoi Sze
Texas A&M University, USA

El-Ghazali Talbi
Université des Sciences & Technologies de Lille, France

Esko Ukkonen
University of Helsinki, Finland

Ugo Vaccaro
Universitá di Salerno, Italy

Gwenn Volkert
Kent State University, USA

Jianxin Wang
Central South University, China

Zidong Wang
Brunel University, UK

Limsoon Wong
NUS, Singapore

Weili Wu
University of Texas at Dallas, USA

Fang Xiang Wu
University of Saskatchewan, Canada

Hongwei Wu
University of Georgia, USA

Dong Xu
University of Missouri, USA

Jack Y. Yang
Harvard University, USA

Mary Qu Yang
National Human Genome Research Institute, USA

Yanqing Zhang
Georgia State University, USA

Xuegong Zhang
Tsinghua University, China

Kaizhong Zhang
University of West Ontario, Canada

Si-qing Zheng
University of Texas at Dallas, USA

Wei-Mou Zheng
Chinese Academy of Sciences, China

Wei Zhong
University of South Carolina, Upstate, USA

Ying Zhu
Georgia State University, USA

External Reviewers

José Augusto Amgarten Quitzau
Irina Astrovskaya
Sudha Balla
Jan Baumbach
Dumitru Brinza
Cedric Chauve
Shihyen Chen
Jaime Davila
Gianluca Della Vedova
Ping Deng
Riccardo Dondi
Lam Fumei
Stefan Gremalschi
Yuanchen He
Inke Hildebrandt
Katharina Jahn
Teemu Kivioja
Deepak Kumar
Phil Lee
Weiming Li
Cui Lin
Chunmei Liu
Jingping Liu
Zhiping Liu
Yi Lu
Praveen Madiraju
Tom Milledge
Julia Mixtacki
Kimmo Palin
Graziano Pesole
Mihail Popescu
Pasi Rastas
Raffaella Rizzi
Baozhen Shan
Mingjun Song
Yinglei Song
Thao Tran
Rui-Sheng Wang
Robert Warren
Roland Wittler
Matthias Wolf
Zikai Wu
Lei Xin
Zhongnan Zhang
Zhongyuan Zhang
Xingming Zhao

Table of Contents

GFBA: A Biclustering Algorithm for Discovering Value-Coherent Biclusters[⋆]

Xubo Fei[1], Shiyong Lu[1], Horia F. Pop[2], and Lily R. Liang[3]

[1] Dept. of Computer Science, Wayne State University, USA
{xubo,shiyong}@wayne.edu
[2] Dept. of Computer Science, Babes-Bolyai University, Romania
hfpop@cs.ubbcluj.ro
[3] Dept. of Computer Science and IT, University of the District of Columbia, USA
lliang@udc.edu

Abstract. Clustering has been one of the most popular approaches used in gene expression data analysis. A clustering method is typically used to partition genes according to their similarity of expression under different conditions. However, it is often the case that some genes behave similarly only on a subset of conditions and their behavior is uncorrelated over the rest of the conditions. As traditional clustering methods will fail to identify such gene groups, the biclustering paradigm is introduced recently to overcome this limitation. In contrast to traditional clustering, a biclustering method produces biclusters, each of which identifies a set of genes and a set of conditions under which these genes behave similarly. The boundary of a bicluster is usually fuzzy in practice as genes and conditions can belong to multiple biclusters at the same time but with different membership degrees. However, to the best of our knowledge, a method that can discover fuzzy value-coherent biclusters is still missing. In this paper, (i) we propose a new fuzzy bicluster model for value-coherent biclusters; (ii) based on this model, we define an objective function whose minimum will characterize good fuzzy value-coherent biclusters; and (iii) we propose a genetic algorithm based method, Genetic Fuzzy Biclustering Algorithm (GFBA), to identify fuzzy value-coherent biclusters. Our experiments show that GFBA is very efficient in converging to the global optimum.

1 Introduction

Clustering has been one of the most popular approaches used in gene expression data analysis. It is used to group genes according to their expression under multiple conditions or to group conditions based on the expression of a number of genes. When a clustering method is used for grouping genes, it typically partitions genes according to their similarity of expression under all conditions. However, it is often the case that some genes behave similarly only on a subset of conditions and their behavior is uncorrelated over the rest of the conditions. Therefore, traditional clustering methods will fail to identify such gene groups.

[⋆] This work was partially supported by the Agricultural Experiment Station at the University of the District of Columbia (Project No.: DC-0LIANG; Accession No.: 0203877).

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 1–12, 2007.

Consider the gene expression data matrix shown in Table 1. If we consider all conditions, genes 1, 2, and 4 do not seem to behave similarly since their expression values are uncorrelated under condition 2 - while genes 1 and 2 have an increased expression value from condition 1 to condition 2, the expression of gene 4 drops from condition 1 to condition 2. However, these genes behave similarly under conditions 1, 3, and 4 since all their expression values increase from condition 1 to condition 3 and increase again under condition 4. A traditional clustering method will fail to recognize such a cluster since the method requires the three genes to behave similarly under all conditions which is not the case.

Table 1. A sample gene expression data matrix and a hidden bicluster

	cond. 1	cond. 2	cond. 3	cond. 4
gene1	**0.0**	5.0	**1.0**	**2.0**
gene2	**1.0**	20.0	**2.0**	**3.0**
gene3	10.0	10.0	20.0	6.0
gene4	**2.0**	0.0	**3.0**	**4.0**

To overcome the limitation of traditional clustering, the biclustering paradigm [1] was introduced recently and several biclustering methods have been developed under this paradigm; see [2] for a recent survey of existing bicluster models and methods. In contrast to traditional clustering, a biclustering method produces biclusters, each of which identifies a set of genes and a set of conditions under which these genes behave similarly. For example, an appropriate biclustering method will recognize highlighted hidden bicluster from Table 1.

However, in practice, the boundary of a bicluster is usually fuzzy for three reasons: (i) the microarray dataset might be noisy and incomplete, (ii) the similarity measurement between genes is continuous and there is no clear cutoff value for group membership, and (iii) a gene might behave similarly to gene A under a set of conditions and behave similarly to another gene B under another set of conditions. Therefore, there is a great need for a fuzzy biclustering method, which produces biclusters in which genes and conditions can belong to a cluster partially and to multiple biclusters at the same time with different membership degrees.

The main contributions of this paper are:

1. We propose a new fuzzy bicluster model for value-coherent biclusters.
2. Based on this model, we define an objective function whose minimum will characterize good fuzzy value-coherent biclusters and facilitate the tradeoff between the number of genes, the number of conditions, and the quality of returned biclusters.
3. We propose a genetic algorithm based method, Genetic Fuzzy Biclustering Algorithm (GFBA), to identify fuzzy value-coherent biclusters. Our experiments show that GFBA is efficient in its converging to the global optimum and produces biclusters with higher quality than existing methods.

Organization. The rest of this paper is organized as follows. In section 2, We introduce the related work on biclustering. In section 3, we formally define a new fuzzy bicluster model and objective function. In section 4, we propose GFBA for discovering biclusters. Experimental results and comparison are analyzed in section 5 and section 6 concludes the paper.

2 Related Work

A recent survey by Madeira and Oliveira [2] identifies four major classes of biclusters: 1) biclusters with constant values, 2) biclusters with constant values on rows or columns, 3) biclusters with coherent values, and 4) biclusters with coherent evolutions.

The first class of biclusters are submatrixes where all values are equal. However, in real datasets, constant biclusters are usually masked by noise. Therefore, a merit function is needed to quantify the quality of constant biclusters. Hartigan [3] defines the variance of a bicluster as such a merit function and proposes a partition-based algorithm to discover constant biclsuters. Given a user specified parameter *K*, the algorithm partitions the original matrix into K biclusters and then calculates the variance for each bicluster. Based on the above variance, Tibshirani et al. [4] propose a permutation-based method to induce the optimal number of biclsuters, K. In addition to using the variance defined by Hartigan [3], Cho et al. [5] also use the total squared residue to quantify the homogeneity of a bicluster. Therefore, their framework can find not only constant biclusters, but also value-coherent biclusters (class 3).

The second class of biclusters are submatrixes with constant values on rows or columns. Getz et al. [6] propose to apply a normalization procedure first to the input matrix before their coupled two-way clustering method is performed. The normalization procedure transforms a row-constant or column-constant bicluster into a constant bicluster (on both rows and columns). To discover row-constant biclusters hidden in noisy data, Califano et al. [7] propose an approach to discover biclusters that for each row, the difference between two extreme values is within some user specified value δ. The same approach can be applied to the discovery of column-constant biclusters. Sheng et al. [8] propose a Bayesian based approach and use either the row-column orientation or column-row orientation to discover column-constant or row-constant biclusters. Finally, Segal et al. [9] introduce a probabilistic model based approach to discover column-constant biclusters, which are more general than the previous approaches.

The third class of biclusters are submatrixes with coherent values on both rows and columns. Cheng and Church [1] define the mean squared residue score to quantify the incoherence of a bicluster and propose a greedy algorithm to discover biclusters with scores lower than some threshold. Yang et al. [10] generalize the definition of δ-bicluster to deal with null values and use the FLexible Overlapped biClustering $(FLOC)$ algorithm to discover a set of biclusters simultaneously. The Coupled Two-Way Clustering $(CTWC)$ proposed by Getz el al. [6] and the Interrelated Two-Way Clustering $(ITWC)$ proposed by Tang et al. [11] are two iterative algorithms based on a combination of the results of one-way clustering on both dimensions. Lazzeroni and Oven [12] introduce the plaid model where the value of an element in the data matrix is viewed as a sum of layers which take into account the interactions between biclusters.

Bleuler et al. [13] propose a EA (evolutionary algorithm) framework that embeds a greedy strategy. Chakraborty et al. [14] use genetic algorithm to eliminate the threshold of the maximum allowable dissimilarity in a bicluster.

The fourth class of biclusters are submatrixes that have coherent evolutions across the rows and/or columns of the data matrix regardless of their exact values. Ben-Dor et al. [15] define a bicluster as an order-preserving submatrix (OPSM) in which the sequence of values in every row is strictly increasing. Liu and Wang [16] generalize the OPSM model by allowing grouping so that columns with insignificant value differences will be considered to have the same ranking and assigned to the same group.

The fuzzy sets theory, developed by Zadeh [17], allows an object to partially belong to one cluster and can belong to more than one clusters at the same time with different membership degrees. As opposed to traditional clustering techniques, fuzzy clustering does not require exclusive membership of data items to a particular class which is advantageous, in that it accommodates uncertainty and imprecision. The most famous fuzzy clustering technique is fuzzy C-means (FCM) algorithm [18] and many other analytic fuzzy clustering approaches are derived from it. A noise clustering (NC) algorithm has been proposed to overcome sensitivity of the FCM algorithms to noisy data [19]. R. Krishnapuram et al.[20] introduce an approach called Possibilistic C Means (PCM) algorithm to overcome the relative membership problem of the FCM.

3 Our Proposed Fuzzy Value-Coherent Bicluster Model and Its Objective Function

In this section, we first give an overview of the value coherent bicluster model proposed by Cheng and Church [1], and then based on this model, we develop our fuzzy value-coherent bicluster model. Through out the whole paper, we will use the notations summarized in Table 2. These notations will be defined later.

Table 2. Notations used in this paper

a_{ij}	the entry in row i and column j
a_{iJ}	the mean of row i of a bicluster
a_{Ij}	the mean of column j of a bicluster
a_{IJ}	the mean of a whole bicluster A_{IJ}
R_{ij}	the residue of a_{ij} in a bicluster
Q_{iJ}	the row sum squared residue of row i
Q_{Ij}	the column sum squared residue of column j
$f_I(i)$	the fuzzy membership value for row i
$f_J(j)$	the fuzzy membership value for column j
$H(I,J)$	the (fuzzy) mean squared residue for bicluster A_{IJ}

3.1 The Value-Coherent Bicluster Model

Consider a gene expression dataset represented by a M by N matrix A, in which rows represent genes, columns represent conditions, and a_{ij} represents the expression of gene i under condition j. Let $G = \{g_1, \cdots, g_M\}$ and $C = \{c_1, \cdots, c_n\}$ be the sets of genes and conditions in A, respectively. Let $I \subseteq G$ and $J \subseteq C$, we use A_{IJ} to denote the submatrix formed by extracting from A all rows and columns in I and J. A (crisp) *bicluster* is a submatrix of A.

Cheng and Church [1] define a value-coherent bicluster model based on an additive model in which each entry a_{ij} is obtained by the sum of the background effect μ, row i's effect α_i, and column j's effect β_j:

$$a_{ij} = \mu + \alpha_i + \beta_j \tag{1}$$

These effects are defined as $\mu = a_{IJ}$, $\alpha_i = a_{iJ} - a_{IJ}$, and $\beta_j = a_{Ij} - a_{IJ}$, where a_{iJ}, a_{Ij}, and a_{IJ} are the means of row i, column j, and the whole bicluster A_{IJ}, respectively, which are defined as follows: $a_{iJ} = \frac{\sum_{j\in J} a_{ij}}{|J|}$, $a_{Ij} = \frac{\sum_{i\in I} a_{ij}}{|I|}$, $a_{IJ} = \frac{\sum_{i\in I}\sum_{j\in J} a_{ij}}{|I|\cdot|J|}$. $\mid I \mid$ and $\mid J \mid$ denote the cardinalities of sets I and J, respectively. As a result, each a_{ij} satisfies the following equation.

$$a_{ij} = a_{iJ} + a_{Ij} - a_{IJ} \tag{2}$$

A bicluster A_{IJ} is *fully value-coherent* if all entries in A_{IJ} satisfy Equation (2).

However, given an arbitrary submatrix A_{IJ}, it might not be a fully value-coherent bicluster. To quantify the value coherence of a bicluster, the notion of *residue* is introduced to calculate the difference between the observed value of a_{ij} and the expected value of a_{ij} if A_{IJ} was a fully value-coherent bicluster.

$$R_{ij} = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \tag{3}$$

Finally, the incoherence of the whole bicluster A_{ij} is defined as the mean squared residue of the bicluster as follows.

$$H(I, J) = \frac{\sum_{i\in I}\sum_{j\in J} R_{ij}^2}{|I|\cdot|J|} \tag{4}$$

Problem Statement. Under this model, a formal statement of *the value-coherent biclustering problem* is as follows: given a gene expression data matrix A and a user provided value δ, return all biclusters A_{IJ} with $H(I, J) \leq \delta$.

3.2 Our Proposed Fuzzy Value-Coherent Bicluster Model

In this section, we extend the above value-coherent bicluster model to a fuzzy model. In contrast to a crisp bicluster, which either contains a gene or a condition completely or does not contain it at all, a fuzzy bicluster can contain a gene or a condition *partially*.

More formally, given a gene expression dataset represented by a M by N matrix A with gene set G and condition set C. Let I be a fuzzy set defined over G with a fuzzy membership function f_I where $0 \leq f_I(i) \leq 1$ for $1 \leq i \leq M$. Similarly, let J be a fuzzy set defined over G with a fuzzy membership function f_J where $0 \leq f_J(j) \leq 1$ for $1 \leq j \leq N$. We use A_{IJ} to denote a fuzzy bicluster that is formed by associating each gene i with a membership value $f_I(i)$ and each condition j with a membership value $f_J(j)$ to reflect the degrees that they belong to the fuzzy bicluster. Therefore, a crisp bicluster a is special case of a fuzzy bicluster. The cardinalities of fuzzy sets I and J are represented as $|I|$ and $|J|$ such that $|I| = \sum_{i=1}^{M} f_I(i)$ and $|J| = \sum_{j=1}^{N} f_J(j)$.

Let $a_{iJ} = \frac{\sum_{j=1}^{N} f_J(j)^m \cdot a_{ij}}{\sum_{j=1}^{N} f_J(j)^m}$, $a_{Ij} = \frac{\sum_{i=1}^{M} f_I(i)^m \cdot a_{ij}}{\sum_{i=1}^{M} f_I(i)^m}$, and $a_{IJ} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f_I(i)^m \cdot f_J(j)^m \cdot a_{ij}}{\sum_{i=1}^{M}\sum_{j=1}^{N} f_I(i)^m \cdot f_J(j)^m}$ be the means of row i, column j, and the whole bicluster A_{IJ}, respectively, m is a user defined value called *fuzziness parameter*, which is used to adjust the power of $f_I(i)$ or $f_J(j)$. The larger the value of m is, the greater the power of $f_I(i)$. We define the residue of an entry a_{ij} as:

$$R_{ij} = a_{ij} - a_{iJ} - a_{Ij} + a_{IJ} \tag{5}$$

We then define the incoherence of the whole fuzzy bicluster A_{IJ} as the mean squared residue of the bicluster.

$$H(I,J) = \frac{\sum_{i=1}^{M}\sum_{j=1}^{N} f_I(i)^m \cdot f_J(j)^m \cdot R_{ij}^2}{|I| \cdot |J|} \tag{6}$$

The reader can verify that if we restrict each fuzzy membership value to be 0 or 1 then Equation (6) is reduced to Equation (4). Therefore, our model is a fuzzy generalization of the basic value-coherent bicluster model.

In addition, we define the row sum squared residue $Q_{iJ} = \sum_{j=1}^{N} f_J(j)^m \cdot R_{ij}^2$ and the column sum squared residue $Q_{Ij} = \sum_{i=1}^{M} f_I(i)^m \cdot R_{ij}^2$ to indicate the contributions of gene i and condition j to the total incoherence, respectively:

3.3 Our Proposed Objective Function

Suppose a user is interested in returning the fuzzy bicluster A_{IJ} with $|I| = \xi_I$ and $|J| = \xi_J$ where ξ_I and ξ_J are two user provided fixed values such that $H(I,J)$ is minimized. The following theorem states one necessary condition for the minimization of $H(I,J)$. This condition can be used in an iterative procedure to update I and J's membership functions to achieve the minimization of $H(I,J)$.

Theorem 1. *Given fixed values $m \in (1,\infty)$, $\xi_I > 0$, and $\xi_J > 0$, $H(I,J)$ with constraints $|I| = \xi_I$ and $|J| = \xi_J$ is globally minimal only if:*

$$f_I(i) = \xi_I \cdot \frac{\frac{1}{Q_{iJ}^{1/(m-1)}}}{\sum_{i=1}^{I} \frac{1}{Q_{iJ}^{1/(m-1)}}} \tag{7}$$

$$f_J(j) = \xi_J \cdot \frac{\frac{1}{Q_{Ij}^{1/(m-1)}}}{\sum_{j=1}^{J} \frac{1}{Q_{Ij}^{1/(m-1)}}} \tag{8}$$

Proof. see [21].

The condition stated in the above theorem can be used to discover biclusters with a given size. To discover biclusters with arbitrary sizes, if one uses the mean squared residue as the objective function, our experiments show that most discovered biclusters will have small sizes (volumes), this is very undesirable since larger biclusters should be more interesting to a biologist as they tend to be more significant. This phenomenon can easily be explained by the definition of the mean squared residue – the larger a bicluster is, the closer its mean squared residue is to the mean squared residue of the whole matrix. Thus smaller biclusters tend to have extreme mean square residues, while larger biclusters tend to have mean square residues that are in the middle.

To solve this problem, inspired by PCM [20], we propose the following objective function.

$$\begin{aligned} H_m(I,J) = H(I,J) \\ &+ \frac{\Sigma_{i=1}^{M}\eta \cdot H(I,J) \cdot (1 - f_I(i))^m}{M - |I|} \\ &+ \frac{\Sigma_{j=1}^{N}\xi \cdot H(I,J) \cdot (1 - f_J(j))^m}{N - |J|} \end{aligned} \tag{9}$$

where the first term is used to control the quality of a bicluster by minimizing its incoherence, the second and third terms are used to promote a bicluster with more genes and more conditions. η and ξ are parameters provided to satisfy different requirements on the incoherence and the sizes of the biclusters. If biologists need biclusters with more genes, they can use greater η value; if they need biclusters with more genes they can increase ξ values. On the other side if they require higher quality they can use smaller values. $H(I,J)$ is the mean squared residue score which is used to adjust the weight of compensation.

Theorem 2. *Given fixed value $m \in (1,\infty)$, $H_m(I,J)$ is globally minimal only if:*

$$f_I(i) = \frac{1}{1 + (\frac{Q_{iJ}\cdot(M-|I|)}{|I||J|\cdot\eta\cdot H(I,J)})^{1/(m-1)}} \tag{10}$$

$$f_J(j) = \frac{1}{1 + (\frac{Q_{Ij}\cdot(N-|J|)}{|I||J|\cdot\xi\cdot H(I,J)})^{1/(m-1)}} \tag{11}$$

Proof. see [21].

It is obvious from (10) and (11) that $f_I(i)$ and $f_J(j)$ lie in the desired range. The genes and conditions that produce large residues will be reassigned with low membership degrees while those co-expressed genes and conditions can get high membership degrees as we expect.

4 Our Proposed GFBA Algorithm

4.1 An Overview of GFBA

Equations (10) and (11) provide an efficient and stable optimization method to minimize the objective function in (9). Unfortunately, it is dependent on initial conditions and might end in a local optimum. In order to ensure the algorithm to converge to a global optimal solution, we propose a genetic algorithm (GA) based fuzzy biclustering algorithm, called GFBA whose pseudocode is outlined in Figure 1, in which g is the number of generations, p is the number of biclusters in each population, mp is the probability of mutation, r is the fraction of the population to be replaced by crossover in each population, cp is the fraction of the population to be replaced by crossover in each population, z is the number of biclusters in each population, and m is the fuzziness parameter. It is different from the normal genetic algorithm in that we have an additional step called *BiclusterOptimization* to speed up the convergence process. Psudocode of BiclusterOptimization is shown in Figure 2.

```
Algorithm GFBA
Input: $g$,$p$,$mp$,$r$,$cp$,$z$,$m$;
Output: Biclusters
Begin
    Initialization()
    For $i$ =0 to $Z - 1$
        Selection()
        Crossover()
        Mutation()
        BiclusterOptimization()
    End For
End Algorithm
```

Fig. 1. Pseudocode of GFBA

```
Algorithm BiclusterOptimization
Input: $< I, J >$;
Output: $< I', J' >$;
Begin
    $I^1 = I, J^1 = J$
    While $||I^{k+1} - I^k|| + ||J^{k+1} - J^k|| < \varepsilon$
        Calculate $H(I, J)$ using (6)
        Update $I^{k+1}$ using (10)
        Update $J^{k+1}$ using (11)
    End While
$I' = I^{k+1}, J' = J^{k+1}$
End Algorithm
```

Fig. 2. Pseudocode of BiclusterOptimization

4.2 Solution Encoding and Fitness Function Definition

As in GA, GFBA maintains a population of coded solutions. A natural way of coding a bicluster is to consider two chromosomes of length M and N representing the genes and conditions. In this case, each allele corresponds to the fuzzy membership degree of a gene or condition. Thus in GFBA, each solution is encoded with a pair of vectors $S_z = (I, J)$ where $I = \{I_1, \cdots, I_M\}(0 < I_i < 1)$ and $J = \{J_1, \cdots, J_N\}(0 < J_j < 1)$. Then we define the fitness function as: $F(S_z) = \frac{1}{H_m(S_z)}$ in which $H_m(S_z)$ can be calculated by (9). We choose the inverse of $H_m(S_z)$ as the fitness value because our goal is to minimize the objective function shown in Equation (9) and thus those biclusters with lower values will be given higher probabilities to survive.

4.3 Operators

Initialization: The genetic algorithm begins with an initial population. In our experiment, the initial population is randomly generated, that is, all the membership degrees are initialized with random numbers between 0 and 1.

Selection: We use Roulette Wheel Selection (RWS), the most commonly used form of GA selection, for the selection operator. When using RWS, a certain number of biclusters $(1 - cp) \cdot z$ of the next generation are selected probabilistically, where the probability of selecting a bicluster S_z is given by

$$Pr(S_z) = \frac{Fitness(S_z)}{\Sigma_{z=1}^{Z} Fitness(S_z)} \tag{12}$$

With RWS, each solution will have a probability to survive by being assigned with a positive fitness value. A solution with a smaller H_m has a greater fitness value and hence has a higher probability to survive. On the other side, weaker solutions also have a chance to survive the selection process. This is an advantage, as though a solution may be weak, it may still contain some useful components.

Crossover and Mutation: Then $cp \cdot z/2$ pairs of parents are chosen probabilistically from the current population and the crossover operator will produce two new offsprings for each pair of parents using one point crossover technique on genes and conditions separately. Now the new generation contains the desired number of members and the mutation will increase or decrease the membership degree of each gene and conditions with a small probability of mutation mp.

Bicluster Optimization: Although the ordinary GA algorithm with above operators may converge and discover biclusters, it will take a lot of time since the initial assignments are random and subsequent evolution process are blind and probabilistic. To solve this problem, We try to leverage the advantage of efficiency and robustness of FCM and proposed BiclusterOptimization function Fig.2. BiclusterOptimization function is a simply Picard iteration through necessary condition $||I^{k+1} - I^k|| + ||J^{k+1} - J^k|| < \varepsilon$. In each generation we use it to update the membership degrees of the genes and conditions resulting from a new generation.

4.4 Interpretation of Fuzzy Biclustering Results

In contrast to traditional biclustering algorithms, our GFBA produces a set of fuzzy biclusters, each of which includes a subset of genes and conditions along with their membership values. For example, given the matrix in Table 3, our algorithm will return the following fuzzy bicluster that is presented by two vectors: Gene(0.99, 0.99, 0.99, 0.99, 0.13) and Con(0.99, 0,99, 0.99, 0.99, 0.81). Each vector indicates the degree of possibility for each gene/condition belonging to the bicluster.

Table 3. A sample gene expression data matrix

	cond.1	cond.2	cond.3	cond.4	cond.5
gene1	2.0	3.0	6.0	1.0	10.0
gene2	3.0	4.0	7.0	2.0	11.0
gene3	5.0	6.0	9.0	4.0	12.0
gene4	6.0	7.0	10.0	5.0	13.0
gene5	40.0	40.0	40.0	40.0	40.0

Fuzzy biclusters provide richer information than regular biclusters as fuzzy biclusters associate with each gene/condition a membership value. However, humans are more familiar with the analysis and reasoning of regular biclusters that are not fuzzy, to accommodate this, our GFBA algorithm provides an optional step to select the most powerful genes and conditions from a fuzzy bicluster. More specifically, given a fuzzy bicluster A_{IJ} and a user specified threshold δ, we can interpret A_{IJ} as a regular bicluster $A_{I'J'}$ as follows.

$$I' = \{i|f_I(i) >= \alpha\} \tag{13}$$

$$J' = \{j|f_J(j) >= \alpha\} \tag{14}$$

Following the above example, if we choose α = 0.8, then the interpretation of the resulting fuzzy bicluster is a regular bicluster that contains genes 1-4 and conditions 1-5 since the membership value of gene 5 is smaller than 0.8 but the membership value for condition 5 is more than 0.8.

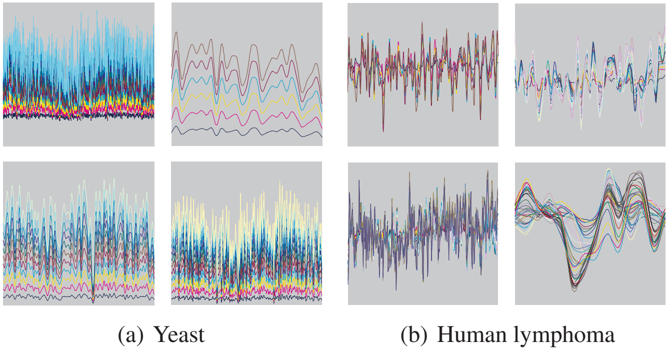
5 Experimental Results

We conducted experiments using GFBA on the same gene expression data sets as used by Y. Cheng and G.M. Church [1]. The yeast Saccharomyces cerevisiae cell cycle expression dataset contains 2,884 genes and 17 conditions. These genes were selected according to Tavazoie et al. (1999). The gene expression values were transformed by scaling and logarithm $x- > 100log(10^5x)$. So the values were mapped into the range 0 and 600 and missing values were represented by -1 in the yeast dataset.

The human lymphoma dataset contains 4026 genes and 96 conditions. The human data was downloaded from the website for supplementary information for the article by Alizadeh et al. (2000). The expression levels were reported as log ratios and after a scaling by a factor of 100, the data values are in the range between -750 and 650, with 47,639 missing values. The matrices introduced above were obtained from http://arep.med.harvard.edu/biclustering

Fuzziness parameter m is important which determines the degree of fuzziness. In general, the larger m is, the "fuzzier" are the membership assignments. Doulaye Dembele. and Philippe Kastner [22] applied traditional FCM on microarray data using yeast and human dataset and found 2 is not a good choice for microarray data. In our experiment we found 1.6 is an appropriate fuzziness parameter for fuzzy biclustering on microarray data. Parameters η and ξ in Eq.22 do not necessary needed as input since we can randomly choose different values for different solutions thus we can find biclusters with different sizes. Parameter ε is used in the procedure BiclusterOptimization as a termination criterion between 0 and 1. In our experiment we choose 0.2.

There are also some parameters used for genetic framework: cp the fraction of the population to be replaced by crossover in each population is 0.7; z the number of biclusters in each population is 300; mp probability of mutation is 0.01; g the number of generations is optional, the higher the better. In our experiment we choose 40. Figure 3 shows Four sample biclusters discovered from yeast expression and human lymphoma expression dataset. The numbers of genes and conditions in each are reported in the format of (residue value, number of genes, number of conditions) as follows: for yeast


(a) Yeast

(b) Human lymphoma

Fig. 3. Sample biclusters discovered by our GFBA algorithm

expression dataset (194.5, 687, 10), (81.9, 143, 6), (95.9, 105, 11), (135.8, 232, 12); for human lymphoma expression dataset (423.4, 120, 6), (539.9, 51, 16), (778.4, 240, 25), (541.8, 14, 39).

Our fuzzy bicluter model is a natural extension of δ-Bicluster [1] and our objective function is a complementarity of fuzzy mean squared residue. It is difficult to compare the quality of different algorithms. Cheng and Church's algorithm is very efficient in finding biclusters with small residues while our algorithm show advantages in fuzziness and flexible and are more capable in discovering large biclusters with relatively small residue. Our objective function can well quantify the quality of biclusters but we can not use it to compare with other algorithm since we are the only ones to use it. Here we use a bicluster quality measure $\frac{1}{n} \cdot \sum_{i=1}^{n} \frac{Residue_i}{Volume_i}$ proposed by A. Chakraborty and H. Maka [14] by calculating the average residue/volume ratio of biclusters. Here n is the number of biclusters returned by an algorithm. Although this metric is better than the mean squared residue, for the control of quality, it is not desirable. This is because a small change of the threshold might affect the number of returned biclusters dramatically, either too many or too few will be returned. However it can be a fair measurement for comparison. On Yeast Dataset, the mean bicluster quality value of our algorithm is 0.02878 while Cheng and Church's algorithm is 1.39978 and Chakraborty's Genetic algorithm 0.05132. On Lymphoma Dataset, our algorithm is 0.109 while Cheng and Church's algorithm is 0.8156 and Chakraborty's Genetic algorithm 0.1247. More detailed results are available at [21].

6 Conclusions and Future Work

In this paper, we proposed an innovative fuzzy bicluster model, in which biclusters can be represented by the degree of possibilities that the genes and conditions belong to these biclusters. Based on this model, we defined an objective function whose minimum will characterize good fuzzy value-coherent biclusters and proposed a genetic algorithm based optimization method. Our experiments showed that our method is very efficient in converging to the global optimum. Future work includes the investigation of methods for discovering other classes of fuzzy biclusters and the applications of these methods to gene expression data analysis.

References

1. Cheng, Y., Church, G.M.: Biclustering of expression data. In: Proc. of the 8th International Conference on Intelligent Systems for Molecular Biology. (2000) 93–103
2. Madeira, S.C., Oliveira, A.L.: Biclustering algorithms for biological data analysis: a survey. IEEE/ACM Transactions on Computational Biology and Bioinformatics **1** (2004) 24–45
3. Hartigan, J.: Direct clustering of a data matrix. Journal of American Statistical Association **67**(337) (1972) 123–129
4. Tibshirani, R., Hastie, T., Eisen, M., Ross, D., Botstein, D., Brown, P.: Clustering methods for the analysis of DNA microarray data. Technical report, Dept. of Health Research and Policy, Dept. of Genetics, and Dept. of Biochemistry, Stanford Univ. (1999)
5. Cho, H., Dhillon, I., Guan, Y., Sra, S.: Minimum sum-squared residue coclustering of gene expression data. In: Proc. of Fourth SIAM Intl Conf. Data Mining. (2004)
6. Getz, G., Levine, E., Domany, E.: Coupled two-way clustering analysis of gene microarray data. In: Proc. of the Natural Academy of Sciences USA. (2000) 12079–12084
7. Califano, A., Stolovitzky, G., Tu, Y.: Analysis of gene expression microarays for phenotype classification. In: Proc. of Intl Conf. Computacional Molecular Biology. (2000) 75–85
8. Sheng, Q., Moreau, Y., Moor, B.D.: Biclustering microarray data by gibbs sampling. Bioinformatics **19** (2003) ii196–ii205
9. Segal, E., Taskar, B., Gasch, A., Friedman, N., Koller, D.: Rich probabilistic models for gene expression. Bioinformatics **17** (2001) S243–S252
10. Yang, J., Wang, W., Wang, H., Yu, P.: Enhanced biclustering on expression data. In: Proc. of 3rd IEEE Conference on Bioinformatics and Bioengineering. (2003) 321–327
11. Tang, C., Zhang, L., Ramanathan, M.: Interrelated two way clustering: an unsupervised approach for gene expression data analysis. In: Proc. of the 2nd IEEE International Symposium on Bioinformatics and Bioengineering. (2001) 41–48
12. Lazzeroni, L., Owen, A.: Plaid models for gene expression data. Technical report, Stanford Univ. (2000)
13. Bleuler, S., Prelic, A., Zitzler, E.: An EA framework for biclustering of gene expression data. In: Proc. of Congress on Evolutionary Computation CEC2004. Volume 1. (2004) 166–173
14. Chakraborty, A., Maka, H.: Biclustering of gene expression data using genetic algorithm. In: Proc. of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. (2005) 1–8
15. Ben-Dor, A., Chor, B., Karp, R., Yakhini, Z.: Discovering local structure in gene expression data: The order- preserving submatrix problem. In: Proc. of the Sixth Int Conf. Computational Biology. (2002) 49–57
16. Liu, J., Wang, W.: OP-Cluster: Clustering by tendency in high dimensional space. In: Proc. of Third IEEE Intl Conf. Data Mining. (2003) 187–194
17. Zadeh, L.: Fuzzy sets. Information and Control **8** (1965) 338–353
18. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum, New York (1981)
19. Dave, R.N.: Characterization and detection of noise in clustering. Pattern Recognition Letters **12**(11) (1991) 657–664
20. Krishnapuram, R., Keller, J.M.: A possibilistic approach to clustering. IEEE Transactions on Fuzzy Systems **1**(2) (1993) 98–110
21. Fei, X., Lu, S., Pop, H.F., Liang, L.: GFBA: A genetic fuzzy biclustering algorithm for discovering value-coherent biclusters, TR-DB-102006-FLPL. Technical report, Dept. of Computer Science, Wayne State Univ. (August 2006) http://paris.cs.wayne.edu/~aw6056/paper.pdf.
22. Dembele, D., Kastner, P.: Fuzzy c-means method for clustering microarray data. **19**(8) (2003) 973–980

Significance Analysis of Time-Course Gene Expression Profiles

Fang-Xiang Wu

Department of Mechanical Engineering and Division of Biomedical Engineering,
University of Saskatchewan, Saskatoon, SK S7N 5A9, Canada
fangxiang.wu@usask.ca

Abstract. This paper proposes a statistical method for significance analysis of time-course gene expression profiles, called SATgene. The SATgene models time-dependent gene expression profiles by autoregressive equations plus Gaussian noises, and time-independent gene expression profiles by constant numbers plus Gaussian noises. The statistical F-testing for regression analysis is used to calculate the confidence probability (significance level) that a time-course gene expression profile is not time-independent. The user can use this confidence probability to select significantly expressed genes from a time-course gene expression dataset. Both one synthetic dataset and one biological dataset were employed to evaluate the performance of the SATgene, compared to traditional gene selection methods: the pairwise R-fold change method and the standard deviation method. The results show that the SATgene outperforms the traditional methods.

Keywords: time-course gene expression, time-dependence, autoregressive model, F-testing.

1 Introduction

With advances in the measurement technology for gene expression and in genome sequencing, it has become possible to simultaneously measure the expression level of thousands of genes in a cell. Time-course gene expression data is obtained from sampled cells at a series of time points over a biological development process, for example, the response of human fibroblasts to serum [1], the response of yeast cells to environmental conditions [2], or the cell division cycle processes [3,4,5]. Such time-course gene expression data provides a dynamic snapshot of most (if not all) of the genes related to the biological development process and may lead to a better understanding of cellular functions. In addition, there is another kind of gene expression data (called non-time course gene expression data) which is acquired for cells from varying conditions (categories), for example, normal cells and tumor cells [6].

In general, the process of acquiring gene expression data are divided three steps [7]: (1) microarray fabrication, (2) sample preparation and array hybridization, and (3) image analysis and numerical data acquisition (called raw data). Numerical gene expression data is usually collected in a data matrix, where each row is the expression profile of a single gene over a variety of conditions (or time points), and each column is the expression values of all genes under a single condition. It is impractical to use

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 13–24, 2007.

raw gene expression data without any pre-processing before further analysis such as cluster analysis, regulatory network analysis [8, 9]. Some basic pre-processing methods include [7]: $\log_2$ transformation, normalization of rows or columns of the data matrix.

Although genes that form the microarray are carefully selected at the stage of microarray fabrication, selecting genes according to their expression data is still critical to get the significant results of any further analysis. Expression data for tens of thousands of genes can be obtained from one single DNA microarray at a time, but not all of these genes are closely related to the biological development process being studied or are of interest. In addition, gene expression data are often contaminated by various noises or “noisy” genes [10]. In general, the most important (or interesting) genes are those which significantly expressed over different conditions. Either excluding genes of interest or including “noisy” genes could degrade the significance of any analysis results. The challenge is how to distinguish genes of interest from a whole set of contaminated gene expression data.

Much attention has paid to select the most interesting genes from non-time-course gene expression data over past years. For gene expression data obtained from a pair of conditions (e.g., normal versus abnormal, or control versus treatment) with multiple replicates, one of the widely used methods in early years is called the R-fold change method [11, 12]. The “R-fold change” method determines a gene to be significantly differentially expressed if the ratio of expression values under two different conditions is greater than R or less than 1/R, where R is a user-preset positive number. However, experimental results [13] showed that R-fold change method could yield an unacceptably high false discovery rate of 73-84% with the values of R between 2 and 4. A re-sampling (bootstrapping) method called SAM [13] was developed to evaluate the statistical significance of gene expression differences under two different conditions with multiple replicates, which significantly improve the false discovery rate, compared to the R-fold change method. Recent work [14] can be viewed as an extension of the SAM.

Two traditional methods are often applied to selecting significantly expressed genes from multi-conditional expression data, such as various stages of a disease process (cancer), growth conditions or tissue or cell types, and time-course gene expression data. One method is an extension of the R-fold change method called the pairwise R-fold change method [15]. A gene is called significantly differentially expressed if a number of pairwise R-fold changes are observed in its expression profile. Computational results [13] showed that the pairwise R-fold change method could also yield a high false discovery rate of 64-71% even if the values of R is between 1.2 and 2.0, and the number of pairwise R-fold changes is 12 out of 16 pairs of conditions. For the log-ratio gene expression data [1, 3, 5, 7], the pairwise R-fold change method is reduced to count the number of time points at which the absolute of gene expression values is larger than a given cut-off value. Another widely used method (here called the standard deviation method) judges a gene to be significantly differentially expressed if the standard deviation of its expression profile is greater than a given cut-off value [1, 7]. In addition, the SAM’s developers pointed [13] that the SAM could be generalized to apply to multi-conditional gene expression data.

Although the traditional methods in [1, 7, 13, 15] could be applied to selecting significantly expressed genes from time-course expression profiles, they did not take the time-dependence of such data into account. For example, arbitrarily permuting time points does not change the results of selection using these traditional methods.

This means that the important information about dynamics in time-course gene expression data is ignored. This study introduces for significance analysis of time-course gene expression profile, called SATgene, which can be used to select the significantly expressed genes from time-course gene expression data. The idea behind this method follows: a true time-course (time-dependent) gene expression profile can be viewed as observations of a dynamic cellular system at a series of time points while a false time-course (time-independent) gene expression profile is a group of random (noisy) observations. If there are two models for true and false time-course gene expression profiles, respectively, distinguishing the significantly expressed genes from noisy ones is reduced to model a selection issue.

This study models a true time-course gene expression profile by an autoregressive model of order p plus Gaussian random noises. There are three reasons why the autoregressive models are used to describe time-course gene expression profiles. Firstly, a living cell can be viewed a dynamic systems which can generally be described by a nonlinear Markov chain model [16] $x_t = f(x_1, \cdots, x_{t-1})$, where x_t is observation values of the system at time point t. An autoregressive model is a good linear approximation of a nonlinear Markov chain model. Secondly, the autoregressive models have recently been used to model time-course gene expression profiles for the purpose of cluster analysis [17, 18]. The results have showed that the autoregressive models are suitable for modeling time-course gene expression data. Finally, this study models a false time-course gene expression profile by a constant number plus Gaussian random noises, which can be viewed as an autoregressive model of order p with some constraints on the coefficients. Thus the standard hypothesis testing for the linear regression model is applicable to determine the statistical significance of the time-dependence.

2 Methods

2.1 Time-Dependent Model

Let $x = \{x_1, \ldots, x_m, \ldots, x_M\}$ be a time series of observation values at equally-spaced time points from a dynamic system. If the model of the dynamic system is known, one could directly check if the time series x is the true observations of the system or just noises. Unfortunately, the model of the dynamic system which produces the time series of observation values is often unknown or is too complicated to be used. In this case, one could model the dynamic system with the observed data from it. One of widely used methods to model time series data is autoregressive analysis [19] which we will adopt to analyze the time dependence of gene expression profiles in this paper. Assume that the value of the time series at time point m depend on the previous p $(\leq m)$ values, then the time-dependent relationships can be modelled by an autoregressive model of order p, denoted $AR(p)$, which is a linear function of the values of previous p observations plus a term representing error. More formally, an autoregressive model of order p may be written as,

$$x_m = \beta_0 + \beta_1 x_{m-1} + \beta_2 x_{m-2} + \cdots + \beta_p x_{m-p} + \varepsilon_m, \quad m = p+1, \ldots, M \tag{1}$$

where β_i $(i = 0, 1, \cdots, p)$ are the autoregressive coefficients, and ε_m $(m = p+1, \cdots, M)$ represent random errors. This study assumes that the errors have a normal distribution independent of time with the mean of 0 and the variance of σ^2. The system of equations (1) can be rewritten in the matrix form as:

$$Y = X\beta + \varepsilon \tag{2}$$

where,

$$Y = \begin{bmatrix} x_{p+1} \\ x_{p+2} \\ \vdots \\ x_M \end{bmatrix}, \ X = \begin{bmatrix} 1 & x_1 & \cdots & x_p \\ 1 & x_2 & \cdots & x_{p+1} \\ \cdots & \cdots & \ddots & \cdots \\ 1 & x_{M-p} & \cdots & x_{M-1} \end{bmatrix}, \ \beta = \begin{bmatrix} \beta_0 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}, \text{ and } \varepsilon = \begin{bmatrix} \varepsilon_{p+1} \\ \varepsilon_{p+2} \\ \vdots \\ \varepsilon_M \end{bmatrix}$$

The likelihood function for model (2) is

$$L(\beta, \sigma^2) = (2\pi\sigma^2)^{-(M-p)/2} \exp\left[-\frac{1}{2\sigma^2} \left\| Y - X\beta \right\|^2 \right] \tag{3}$$

If the rank (X) = $p+1$ holds, it has proved [20] that the maximum likelihood estimates of β and σ^2 are

$$\hat{\beta} = (X^T X)^{-1} X^T Y \tag{4}$$

and

$$\hat{\sigma}^2 = \left\| Y - X\hat{\beta} \right\|^2 /(M - p) \tag{5}$$

respectively. The maximum values of the likelihood is given by

$$L(\hat{\beta}, \hat{\sigma}^2) = (2\pi\hat{\sigma}^2)^{-(M-p)/2} e^{-(M-p)/2} \tag{6}$$

In model (2), the matrix X has $p+1$ columns and $M - p$ rows. Thus a necessary condition for rank (X) = $p+1$ is $M - p > p+1$ or $p < (M-1)/2$.

2.2 Time-Independent Model

For a group of observation values which are not produced by the dynamic systems under consideration, but noisy (random) data, one can simply model them by a constant number plus random errors. Let $x = \{x_1, \ldots, x_m, \ldots, x_M\}$ be a series of time-independent (random) observations. In agreement with model (2), the last ($M - p$) observations can be modelled by

$$x_m = \beta_0 + \varepsilon_m, \ m = p \cdots, M \tag{7}$$

where β_0 is a constant number and ε_m ($m = p, \cdots, M$) are the random errors which are subject to a normal distribution independent of time with mean 0 and variance σ^2. The likelihood function for model (7) is

$$L(\beta_0, \sigma^2) = (2\pi\sigma^2)^{-(M-p)/2} \exp\left[-\frac{1}{2\sigma^2} \sum_{m=p+1}^{M} (x_m - \beta_0)^2 \right] \tag{8}$$

The maximum likelihood estimates of β_0 and σ^2 are

$$\hat{\beta}_0 = \frac{1}{M-p} \sum_{m=p+1}^{M} x_m \tag{9}$$

and

$$\hat{\sigma}_c^{\ 2} = \frac{1}{(M-p)} \sum_{m=p+1}^{M} (x_m - \hat{\beta}_0)^2 \tag{10}$$

respectively. The maximum values of the likelihood is given by

$$L(\hat{\beta}_c, \hat{\sigma}_c^2) = (2\pi\hat{\sigma}_c^2)^{-(M-p)/2} e^{-(M-p)/2} \tag{11}$$

where $\hat{\beta}_c$ is a (*p*+*1*)-dimensional vector whose first components is $\hat{\beta}_0$ and others are zeros.

2.3 Hypothesis Testing

Actually, the time-independent model (7) is also an autoregressive model with the order of zero and can be viewed as model (1) with constraints $\beta_i = 0 \ (i = 1, \cdots, p)$. These constraints can be rewritten in matrix form

$$A\beta = 0 \tag{12}$$

where

$$A = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

The likelihood ratio of model (7) to model (1) is given by

$$\Lambda = \frac{L(\hat{\beta}_c, \hat{\sigma}_c^2)}{L(\hat{\beta}, \hat{\sigma}^2)} = \left(\frac{\hat{\sigma}^2}{\hat{\sigma}_c^2} \right)^{(M-p)/2} \tag{13}$$

As model (7) can be viewed as a regression model (1) with the constraints (8), and the maximum likelihood method is used to obtained $\hat{\sigma}^2$ and $\hat{\sigma}_c^2$, the inequality $\hat{\sigma}^2 \leq \hat{\sigma}_c^2$ comes true. According to the likelihood principle [20], if Λ is too small, model (1) is preferable, i.e. the series $x = \{x_1, \ldots, x_m, \ldots, x_M\}$ is more likely time-dependent than time-independent. Although Λ is not a convenient test statistic, it has proved [20] that

$$F = \frac{M-2p-1}{p} (\Lambda^{-2/(M-p)} - 1) = \frac{M-2p-1}{p} \left(\frac{\hat{\sigma}_c^2}{\hat{\sigma}^2} - 1 \right) \tag{14}$$

has an $F_{p,(M-2p-1)}$ distribution when model (7) is true for a series of observation. When F is too large, the model (7) is rejected, i.e., observation series $x = \{x_1, \ldots, x_m, \ldots, x_M\}$ is time-dependent. Form formulae (14), one can calculate the confidence probability (significant level) that a series of observation is not time-independent. As the regression degree in model (1) is unknown, the significant levels are calculated by formulae (14) for all values d $(1 \le d \le (M-1)/2)$. The proposed SATgene calls a gene to be significantly expressed (time-dependent) if one of these confident levels calculated from its expression profile is larger than a user-preset value.

3 Evaluation

Many factors could affect gene expression profiles during data collection process. Genes related to a process may not be sufficiently expressed, and thus their expression profile may not appear time-dependent. Hence, biological data can only provide anecdotal evidence. This study employs both a synthetic dataset and a biological dataset to investigate the performance of the proposed SATgene, compared with the traditional methods: the standard deviation method and the pairwise R-fold change method.

3.1 Datasets

Synthetic dataset (SYN): A synthetic dataset is generated by the sine function modeling cyclic behaviour of genes expression employed by Schliep [7] and Yeung, et al. [21]. Let x_{ij} be the simulated expression (log-ratio) values of gene i at time point j in the dataset and be modeled by

$$x_{ij} = \alpha_i + \phi(i, j) + d * \varepsilon_{ij} \qquad (15)$$

where α_i represents the average expression level of gene i, and which could be set to zeros if one assumes to shift the mean of each gene expression profile to zero. $\phi(i, j) = \sin(2\pi(j-1)/12 - w_i)$ represents cyclic behaviour of gene expression. Each cycle is assumed to span 14 time points. w_i represents a phase shift for gene i, which is chosen according to the uniform distribution on interval $[0, 2\pi]$. The random variable ε_{ij} represents the observation noise of gene i on array (or at time point) j, which is chosen according to the standard normal distribution. d is a constant number and equal to 0.5 for creating time-dependent gene expression profiles. Using model (15), a time-dependent observation dataset (TDD) was created,

Table 1. Statistics of the synthetic dataset

	TDD	TID
Maximum value	2.4738	3.1398
Minimum value	-2.5675	-3.1060
Average value	0.0098	0.0019
Maximum std	1.2455	1.4374
Minimum std	0.5020	0.4363
Average std	0.8672	0.8614

which contains 300 gene expression profiles with 14 equally-spaced time points. The model (15) with $\phi(i, j)=0$ and $d = 0.9$ was used as the model for creating a time-independent (noisy) observation dataset (TID) which contains 300 random series with 14 time points. Considering that the sine function $\phi(i, j)$ also contributes the standard deviation to gene expression profiles in TDD, the value of $d = 0.9$ was chosen for creating profiles in TID so that the averages of standard deviation of profiles in both TDD and TID are approximately equal. TDD and TID were made up the synthetic dataset SYN. Some statistics of the synthetic dataset is listed in Table 1, where the rows through 2nd to 4th are maximum, minimum, average of gene expression values in TDD and TID, respectively, and the rows through 5th to 7th are maximum, minimum, average of the standard deviation of gene expression profiles. These corresponding statistics for TDD and TID are much close, in particular, the averages of the standard deviation of profiles. Figure 1 (a) and (b) plot profiles of synthetic gene expression in TDD and TID, respectively. One can see that profiles in TDD and TID look similar.

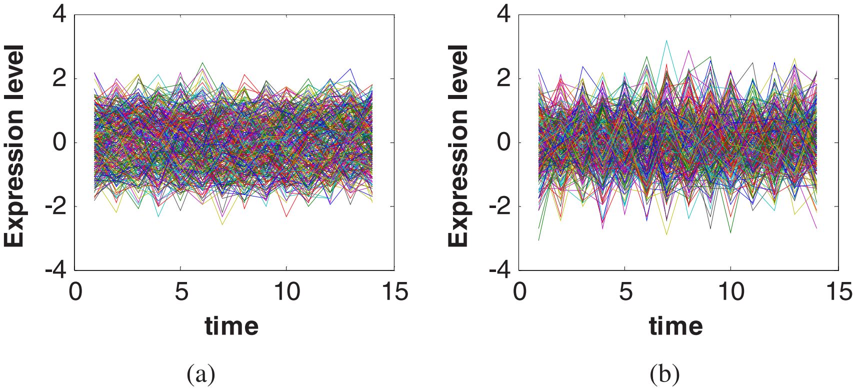


Fig. 1. Profiles of synthetic gene expression: (a) for TDD (b) for TID

Bacterial cell cycle (BAC): This dataset contains gene expression measurements during the bacterial cell cycle division process for about 3000 predicted open reading frames, representing about 90% of all bacterium Caulobacter crescentus genes [5]. The measurements were taken at 11 equally-space time points over 150 minutes. The dataset is publicly available from the website http://caulobacter.stanford.edu/CellCycle. Clones with missing data were excluded in this study. The resultant dataset contains the expression profile of 1594 genes, out of which, 58 genes were previously characterized to be cell cycle-regulated [5].

3.2 Benchmarking Results on SYN

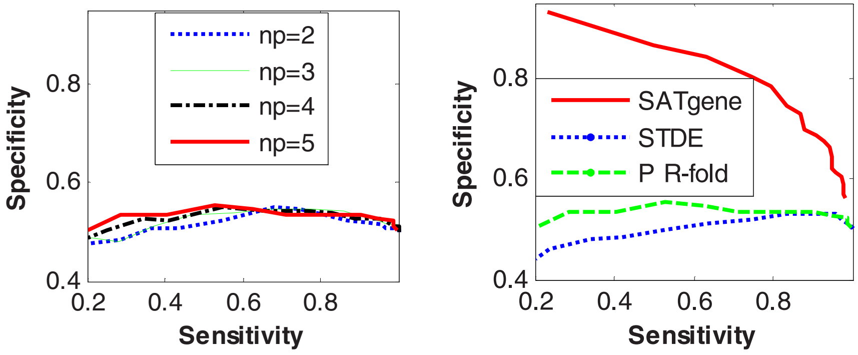
The synthetic dataset YSN created above is used to make a benchmark of the SATgene while the pairwise R-fold fold change methods [15], and the standard deviation method [1, 7] is used to supply a base line. The performance of all three methods is evaluated with two parameters widely accepted and used in the course of method evaluation: sensitivity and specificity [22]. Sensitivity is the ratio of the number of profiles correctly determined to be time-dependent over the number of all true time-dependent profiles, while the specificity is defined as the ratio of the number of profiles correctly determined to be time-dependent over the total number of profiles determined to be time-dependent. The definition for sensitivity is:

$$Sensitivity = \frac{TP}{TP + FN} \tag{16}$$

and that for specificity

$$Specificity = \frac{TP}{TP + FP} \tag{17}$$

respectively, where TP denotes the number of true positives (profiles in TDD determined to be time-dependent by a method), FN denotes the number of false negatives (profiles in TDD determined to be time-independent by a method), and FP denotes the number of false positives (profiles in TID determined to be time-dependent by a method). In general, specificity is small while sensitivity is large, and vice versa. A good method is expected to have both a high specificity and a high sensitivity at a point.



(a) (b)

Fig. 2. Plots of specificity with respect to sensitivity for three methods: (a) for the pairwise R-fold change methods with np =2, 3, 4, 5; (b) for the SATgene, the standard deviation (STDE) method, and the pairwise R-fold (P R-fold) change method with np =5

Let np be the number of time points at which the absolute of expression values is larger than a given cut-off value. Given a value of np, specificity and sensitivity are calculated over a variety of cut-off values for the pairwise R-fold change method. Figure 2(a) depicts the plots of specificity with respect to sensitivity for the pairwise R-fold change method with the various values of np. It is intuitive that the more expression values in a gene expression profile are larger than a given cut-off value, the more significantly expressed the corresponding gene is. Therefore, with increase of np, the pairwise R-fold change method should perform better. However, when the values of np is too large, there are no profiles that have np values larger than a given cut-off value, and thus the pairwise R-fold change method makes no sense. In this study, the maximum number of np is 5 for dataset SYN. Figure 2(a) shows the performances of the pairwise R-fold change methods over various values of np are comparable in terms of sensitivity and specificity on dataset SYN. The fact that the curves in Figure 2(a) are closed to the straight line $specificity = 0.5$ indicates that when the pairwise R-fold change methods are applied to select the time-dependent gene expression profiles, almost half of profiles in the resultant dataset are random profiles. Therefore the pairwise R-fold change methods can not significantly distinguish time-dependent gene expression profiles from time-independent (random) ones.

Figure 2(b) depicts the plots of specificity with respect to sensitivity for the proposed SATgene, the standard deviation method, and the pairwise R-fold change method (represented by the case of $np = 5$). Specificity and sensitivity for the SATgene are calculated over a variety of significance levels calculated by formulae (14). With increase of significance levels, the sensitivity decreases while the specificity increases. Specificity and sensitivity for the standard deviation method are calculated over a variety of standard deviation cut-off values. Obviously the SATgene is the best one while the standard deviation method is the poorest one. For example, at the sensitivity of 20%, the specificity is about 95% for the SATgene, and is much larger than 45% for the standard deviation method and 50% for the pairwise R-fold change method. In an acceptable sensitivity of 80%, the specificity is about 80% for the SATgene, but about 50% for both the standard deviation method and the pairwise R-fold change method. The fact that the curve for the standard deviation method in Figure 2(b) is below the line $specificity = 0.5$ indicates that when the standard deviation method is applied to select time-dependent gene expression profiles, more than half of profiles in the resultant dataset are random profiles.

3.3 Results on Biological Data BAC

In this dataset, there is no *a priori* information for all gene expression profiles whether they are time-dependent or not, but 58 genes in this dataset were previously characterized to be cell cycle-regulated [5]. The expression profiles of these genes should be time-dependent if the whole procedure for data acquisition is ideal. Unfortunately, this procedure is often contaminated by various noises in reality [10]. However, this study employs expression profiles of these 58 genes as control (reference) ones to compare the performance of the SATgene, the standard deviation method, and the pairwise R-fold change method with $np = 2,3,4$.

Table 2. The number of control genes with respect to the confidence probability

Confidence level (%)	50	70	85	90	95	96	97	98	99	99.5	99.7	99.9
# of control genes	55	54	52	50	49	48	47	45	39	29	25	12

All these methods except for the SATgene do not provide the confidence probability that a gene expression profile is not a random profile. The number of control genes with respect to the confidence probability from the SATgene is listed in Table 2. One can see that the number of control genes decreases with increase of the confidence probability.

Figure 3 plots the number of time-dependent genes in BAC with respect to the number of time-dependent control genes determined by various methods. From Figure 3(a), the pairwise R-fold change methods with various values of *np* determined the almost same numbers of time-dependent genes in BAC, given the same number of control genes determined. Comparing with Table 2, one can conclude that the performances of the pairwise R-fold change methods with various values of *np* are comparable in term of the number of genes determined to be time-dependent with respect to the significance level (especially for higher significance level portion). This result is in agreement with that for dataset SYN.

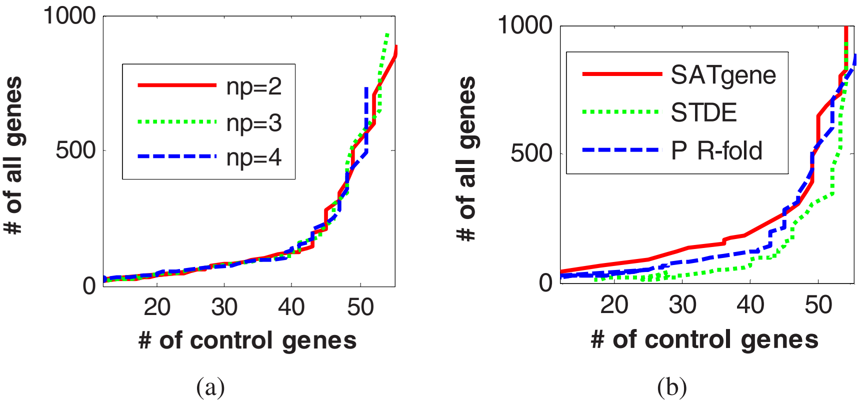


Fig. 3. Plots of the number of time-dependent genes in BAC with respect to the number of time-dependent control genes determined by various methods: (a) for the pairwise R-fold change method with *np* =2, 3, 4; (b) for the SATgene, the standard deviation method (STDE), and the pairwise R-fold (P R-fold) change method with *np* =2

Figure 3 (b) compares the SATgene with the standard deviation method and the pairwise R-fold change method (represented by the case of *np* =2). Obviously the SATgene can find more time-dependent profiles than other two traditional methods while the standard deviation method find the least time-dependent profiles, given the same number of control genes found respectively by these methods. Compared with Table 2, although at the lower confidence level the SATgene and the pairwise R-fold change method are comparable, at higher confidence level the SATgene is clearly better than the pairwise R-fold change method. Over all confidence levels, the standard deviation method is always poorer than the SATgene. Again these results for dataset BAC are in agreement with those for dataset SYN.

4 Conclusion

This paper presents a simple statistical approach to analyze the significance of time-course gene expression profiles and thus select significantly expressed genes from time-course gene expression datasets. The most important feature of the proposed method is that it takes into consideration the inherent time dependence (dynamics) of time-course gene expression patterns. In addition, the presented method is intuitively understandable. Computational experiments on both a synthetic dataset and a biological dataset show that the proposed SATgene significantly improve the traditional methods such as the standard deviation method and the pairwise R-fold change methods.

Time-course gene expression data could provide important information about the biological process from which data is observed, and thus may lead to a better understanding of gene regulatory relationships and further gene regulatory networks. However, traditional gene selection methods may inefficiently use or misuse the information contained in this kind of data. With the superior performance of SATgene over the traditional methods, some directions of future work are to infer gene regulatory relationships and gene regulatory networks with the resultant dataset selected by the SATgene.

Acknowledgment. This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC).

References

1. Iyer, V. R., et al.: The transcript-ional program in the response of human fibroblasts to serum. Science 283(1999) 83-87
2. Gasch, A. P., et al.: Genomic expression programs in the response of yeast cells to environmental changes. Molecular Biology of the Cell 11(2000) 4241-4257
3. Spellman, P. T., et al.: Comprehensive identification of cell cycle-regulated genes of the Yeast Saccharomyces cerevisiae by microarray hybridization. Molecular Biology of the Cell 9 (1998) 3273-3297
4. Whitfield M. L., et al.: Identification of genes periodically expressed in the human cell cycle and their expression in tumors. Molecular Biology of the Cell 13(2002) 1977-2000
5. Laub, M. T., et al: Global analysis of the genetic network controlling a bacteria cell cycle. Science 290(2000) 2144-2148
6. Golub, T. R., et al.: Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science 286(1999) 531-537
7. Eisen, M. B., et al: Cluster Analysis and Display of Genome-Wide Expression Patterns. Proc Natl. Acad. Sci. USA 95 (1998) 14863-14868
8. Bar-Joseph. Z.: Analyzing time series gene expression data. Bioinformatics 20 (2004) 2493-2503
9. Schliep, A., et al.: Analyzing gene expression time-course. IEEE/ACM Trans. on Computational Biology and Bioinformatics 2(2005) 179-193
10. Baldi, P. and Hatfield, G. W.: DNA Microarrays and Gene Expression: From Experiments to Data Analysis and Modeling, Cambridge University Press (2002)

11. Alizadeh, A. A., et al.: Distinct types of diffuse large B-cell Lymphoma identified by gene expression profiling. Nature, 403 (2000) 503-511.
12. Claverie, J. M.: Computational methods for the identification of differential and coordinated gene expression. Human Molecular Genetic 8(1999) 1821–1832
13. Tusher, V. G., et al.: Significance analysis of microarrays applied to the ionizing radiation response. Proc. Natl. Acad. Sci. USA 98 (2001) 5116-5121
14. Smyth, G. K., Michaud, J., and Scott, H. S.: Use of within-array replicate spots for assessing differential expression in microarray experiments. Bioinformatics 21(2005) 2067-2075
15. Ly D. H., et al.: Mitotic misregulation and human aging. Science 287(2000) 2486-2492
16. Melnik, R. V. N.: Dynamic system evolution and Markov chain approximation. Discrete Dynamics in Nature and Society 2(1998) 7-39
17. Ramoni, M. F., et al.: Cluster analysis of gene expression dynamics. Proc. Natl. Acad. Sci. USA 99 (2002) 9121-912.
18. Wu F. X, Zhang W. J., and Kusalik A. J.: Dynamic model-based clustering for time-course gene expression data. Journal of Bioinformatics and Computational Biology 3 (2005) 821-836
19. Harvey, A. C.: Time Series Models, Cambridge: MIT Press (1993)
20. Seber, G. A. F and Lee, A. J.: Linear Regression Analysis, 2nd Edition. Hoboken, N. J.: Wiley Interscience (2003)
21. Yeung K. Y.: (2001) Model-Based clustering and data transformations for gene expression data. Bioinformatics 17(2001) 977-987
22. Baldi, P. and Brunak, S: Bioinformatics: The Machine Learning Approach, 2nd Edition, Cambridge: The MIT Press (2001)

Data-Driven Smoothness Enhanced Variance Ratio Test to Unearth Responsive Genes in 0-Time Normalized Time-Course Microarray Data

Juntao Li, Jianhua Liu, and R. Krishna Murthy Karuturi*

Genome Institute of Singapore, 60 Biopolis ST, Republic of Singapore
{lij9,liujh,karuturikm}@gis.a-star.edu.sg
http://www.gis.a-star.edu.sg

Abstract. Discovering responsive or differentially expressed genes in time-course microarray studies is an important step before further interpretation is carried out. The statistical challenge in this task is due to high prevalence of situations in which the following settings are true: (1) none or insufficiently fewer repeats; (2) 0-time or starting point reference; and, (3) undefined or unknown pattern of response. One simple and effective criterion that comes for rescue is smoothness criterion which assumes that a responsive gene exhibits a smooth pattern of response whereas a non-responsive gene exhibits a non-smooth response. Smoothness of response may be gauranteed if the expression is sufficiently sampled and it can be measured in terms of first order or serial autocorrelation of gene expression time-course using Durbin-Watson (DW) test. But, the DW-test ignores variance of the response which also plays an important role in the discovery of responsive genes while variance alone is not appropriate because of nonuniform noise variance across genes. Hence, we propose a novel *Data-driven Smoothness Enhanced Variance Ratio Test (dSEVRaT)* which effectively combines smoothness and variance of gene expression time-course. We demonstrate that dSEVRaT does significantly better than DW-test as well as other tests on both simulated data and real data. Further, we demonstrate that dSEVRaT can address both 0-time normalized data and the other data equally well.

1 Introduction

Time-course gene expression studies provide valuable information on the dynamics of cellular response to a stimulus from transcription point of view. It is important in the study of evolution of molecular response of the cells to a stimulus and the associated gene regulatory networks [1]. Discovering responsive or differentially expressed genes in such studies is the first and foremost important step before further analysis and interpretation of the data are carried out. Several methods have been proposed for this purpose which can broadly

* Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 25–36, 2007.

be classified as (1) clustering methods; (2) neighborhood methods; (3) pattern based methods; (4) model based methods; and, (5) summarization methods.

Clustering methods [10,11] have been widely used to identify responsive genes. These methods depend on the principle of co-regulation i.e. coregulated genes show similar expression profiles and cluster together. Among various clusters produced by a clustering method, the researcher chooses clusters of interest and interprets the genes belonged to them as responsive genes. These methods are prone to higher error rates because of the lack of gene specific quantification of responsiveness.

Neighborhood methods [3][4][7] are similar to clustering methods except that they quantify each gene's significance based on the number of similarly expressed genes called neighbors they have in the expression space. Friendly neighbors algorithm [3] was used to identify responsive genes for single class case and DiffFNs [4] & MARD [7] are devised for multi-class differential response case. Similar to clustering methods, these methods also rely mostly on the neighborhood of the gene to assess its significance and genes with smaller neighborhood may be rejected as unresponsive genes. This may not be suitable in several cases.

Pattern based methods [22][6][18][14] are especially useful if the pattern of response is pre-defined. These patterns may be parametric or nonparametric. For example, a cosine pattern or fitt [6] or periodic splines [14] is used to identify responsive genes in cell cycle experiments [22][20] and a definition of nonparametric up-down patterns may be used in other experiments [18]. These methods require prior definition of the pattern [6][14][18] and may require long time-course and/or sufficient number repeats at each time point [18].

Model based methods [23][21][15][17][25] are similar to pattern based methods except that pattern of response is not required to be defined apriori. Instead, they fitt linear [21][15][17] model or polynomial model or splines model [23][25] to the gene's expression response. The main problem with these methods is that they require repeats at each time point. In the absence of sufficient number of repeats, their performance degrades. These are especially useful to identify multiclass differentially responsive genes with well repeated experiments.

Summarization methods [24][19][5] aim to summarize certain characteristic relevant to responsiveness such as signed area under response curve and slope of the response curve as in SAM [24], autocorrelation of response as in PEM and DW tests [19][5]. These methods are simple and effective, yet make very weak assumptions on responsiveness and can tolerate the lack of sufficient number of repeats. But, their performance is sensitive to 0-time or starting point normalization; in 0-time normalization, the expression at the first time-point of the time-course which signifies the expression before stimulus or treatment is substracted from the expression at all time points of the gene. This is common in several microarray experiments [11][9][26].

In this paper we address the problem of discovering significantly responsive genes in the following difficult setting: (1) none or insufficiently few repeats; (2) 0-time or starting point reference; (3)undefined or unknown pattern of response; and, (4) sufficiently sampled time-course. Accurate discovery of responsive genes is greatly hampered by insufficiently few or no repeats owing to the cost of microarrays or technical difficulties of obtaining appropriate amount of samples for replicate experiments. The absence of parallel reference time-course using unstimulated or untreated samples is common which requires the use of response just prior to the stimulation or initial observation or 0-time point or starting point of the time-course as reference for the entire time-course for each gene. Due to no replicates and high variance of noise at the reference measurement, this normalization may introduce spurious average level for the time-course which increases error rates of the responsive gene prediction. This can be illustrated as follows: Let the expression of a gene g_i at time T_t be E_{it} for $t \in \{0, 1, 2, ..., n\}$, $i \in \{1, 2, ..., N\}$; E_{it} is a random variable with mean m_{it} (true expression of g_i at T_t) and variance σ_i^2. Then, after 0-time normalization, E_{it} becomes $\widetilde{E_{it}} = E_{it} - E_{i0}$. If g_i is not responsive, then m_{it} is same as m_{i0} and the expected value of the mean of the time-course will also be m_{i0}. But, because of 0-time normalization, expected value of $\widetilde{E_{it}}$ will be $E_{i0} - m_{i0}$ which may not be 0 and the expected value of the mean of the time-course may also be non-zero i.e. $\frac{n-1}{n}(E_{i0} - m_0)$. This may lead to, though variable from gene to gene, gross misinterpretation of the data. The unknown pattern of response coupled with these two factors poses an important statistical challenge in identifying responsive genes in these datasets. Further, a lot of unreplicated time-course microarray datasets are available in the public repositories whose analysis provide good information in the future studies.

If the time-course is sufficiently long, which is true in several time-course studies, we can use smoothness or autocorrelation criterion. Smoothness criterion assumes that a responsive gene exhibits a smooth or autocorrelated pattern of response whereas a non-responsive gene exhibits a non-smooth or non-autocorrelated response about average response. Smoothness can be measured in terms of first order or serial autocorrelation of the signal using Durbin-Watson (DW) test. But, the DW-statistic ignores variance of the response which also plays an important role in the discovery of responsive genes while variance alone is not effective since the noise variance is not uniform across genes. To address this problem, we propose a novel test called *Data-driven Smoothness Enhanced Variance Ratio Test (dSEVRaT)* which efficiently combines smoothness and variance of signal. We demonstrate that dSEVRaT does significantly better than DW as well as other tests on both simulated data and the real data. Further, we demonstrate that dSEVRaT can address both 0-time normalized data and the other data equally well. dSEVRaT, like its predecessors DW and PEM tests, assumes the sampling of expression is sufficient so as to satisfy smoothness and error is symmetric with mean zero (0). These two assumptions are weak enough and hence satisfied in most of the practical time-course datasets.

The remaining part of the paper is organized as follows: Section 2 presents the basic definition of dSEVRaT and its variation to include data dependent parameters suitable for microarray data analysis. Section 3 presents the comparative performance of dSEVRaT with that of PEM and DW on simulated data as well as the real data. Section 4 provides discussion and future directions.

2 Data-Driven Smoothness Enhanced Variance Ratio Test (dSEVRaT)

Let the expression of a gene g_i at time point T_t is denoted by E_{it}. Let $\Delta E_{it} = E_{it} - E_{it-1}$, difference between expression at time T_t and $T_{(t-1)}$. Let the $V(E_i)$ and $V(\Delta E_i)$ denote the variances of $E_i = \{E_{i1}, E_{i2}, ..., E_{in}\}$ and $\Delta E_i = \{\Delta E_{i2}, \Delta E_{i3}, ..., \Delta E_{in}\}$ respectively. Then the variance ratio statistic ($VRaT$) for g_i is defined as

$$VRaT_i = \frac{2V(E_i)}{V(\Delta E_i)}$$

$VRaT_i$ measures the first order autocorrelation or serial correlation of the time series E_i and it asymptotically converges to twice the inverse of Durbin-Watson statistic on $E_i - \mu_i$ where μ_i is average of E_i. The proof of this is simple and straightforward.

$VRaT_i$ does not account for the variance of the signal E_i which plays an important role in the selection of responsive genes. But, variance alone can not effectively discriminate responsive genes from that of unresponsive genes because of different noise variances in the expression measurements of different genes. Hence, we enhance $V(E_i)$ in $VRaT_i$-dependent manner since $VRaT_i$ is invariant of scaling and it is indication of smoothness. The smoothness enhanced $VRaT_i$ ($SEVRaT_i$) is defined as

$$SEVRaT_i = \frac{2V^{Y_i}(E_i)}{V(\Delta E_i)}$$

$$\text{where } Y_i = f(VRaT_i)$$

We call $f(.)$ as *smoothness enhancer function* which has to satisfy the following so called *enhancer constraints*: (1) Continuity constraint i.e. $f(x)$ is differentiable for all $x > 0-$; (2) Basal constraint i.e. $f(.) \geq 1$; and (3) Non-decreasing $f(.)$ i.e. $\forall a, b \geq 0$, $a \leq b \Longrightarrow f(a) \leq f(b)$ or $df(x)/dx \geq 0, \forall x \geq 0$.

The enhancer constraints make sure that the amount of enhancement is non-decreasing with $VRaT_i$ and $SEVRaT_i \geq VRaT_i$ under the assumption of $V(E_i) > 1$. Several forms of $f(.)$ are possible in practice. For the purpose of this paper, we define $f(.)$ as

$$Y_i = f(VRaT_i) = 1 + VRaT_i$$

The above definition satisfies the enhancer constraints since $VRaT_i \geq 0$.

2.1 Data-Driven SEVRaT (dSEVRaT)

$SEVRaT_i$ is the statistic defining the responsiveness of g_i independent of the other genes in the dataset. But, in reality, we have two difficulties: (1) the assumption $V(E_i) > 1$ may be violated for a responsive gene g_i; and (2) some weak signals may turn out to be extremely smooth (as in differential expression analysis of non-time course expression data) leading to higher false discovery rate owing to huge multiple hypothesis testing in microarray analysis. We provide the following solutions for these problems:

Relative Variance Preserving Normalization: Several responsive genes in a dataset (or experiment) may have $V(E_i) \leq 1$ which leads to the attenuation of variance instead of its enhancement. Hence it reduces the true positive rate or recall. To circumvent this problem, we propose to scale all genes' expression measurements by the $K(< 100)\%$ of the standard deviation of the least varying gene in the dataset which gaurantees the validity of the first assumption. This normalization does not change the relative variance of any pair of genes and hence the relative ranking by $V(E_i)$ or $VRaT_i$ which is very important for this analysis and final ranking of the genes. This justifies the name.

Fudge factor in SEVRaT: A typical problem in microarray analysis, especially with insufficient number of samples which is common, is the presence of several data dependent low varying unresponsive genes which can mislead the traditional statistical tests. This problem was addressed in the design of SAM (*Significance Analysis of Microarrays*) procedure [24] by incorporating the S_0 called *fudge factor* to offset lowly varying genes relatively more compared to the highly varying genes. The same idea has been adopted in several other procedures later on [19][12]. We also adopt the same idea for our SEVRaT to offset the low varying and extremely smooth genes as compared to the high variance genes. So we add fudge factor S_0 to the denominator of SEVRaT, $V(\Delta E_i)$; S_0 is user defined parameter, we propose it to be ninety fifth percentile of all $V(\Delta E_i)$.

Thus, incorporating the above data dependent parameters in to $SEVRaT_i$ leads to the so called Data-driven $SEVRaT_i$ ($dSEVRaT_i$) defined as

$$dSEVRaT_i = \frac{2\left(\frac{V(E_i)}{V_{min}}\right)^{Y_i^*}}{\left(\frac{V(\Delta E_i)+S_0}{V_{min}}\right)}$$

$$\text{where } V_{min} = \left(\frac{K}{100}\right)\left(\overset{\textbf{MIN}}{i}\ V(E_i)\right) \text{ and } S_0 \text{ is fudge factor}$$

$$Y_i^* = f(VRaT_i^*) \text{ where } VRaT_i^* = \frac{2V(E_i)}{V(\Delta E_i) + So}$$

2.2 Testing for Statistical Significance

False discovery rate [2][13] and local false discovery rate [13] are used as measures of statistical significance of the selected genes. They are assessed using label permutation procedure as in [24]. False discovery rate of g_i is calculated as a product of estimated proportion of non-differentilly expressed or unresponsive genes in the dataset and fraction of genes in the label permuted data have dSEVRaT more than $dSEVRaT_i$. Local false discovery rate of g_i is calculated as a product of estimated proportion of unresponsive genes and fraction of genes in the label permuted data has $dSEVRaT \in [dSEVRaT_i - \delta, dSEVRaT_i + \delta]$, where δ is bandwidth of dSEVRaT density estimation. Proportion of unresponsive genes is estimated as half of the fraction of genes in the original data have dSEVRaT within 25^{th} percentile to 75^{th} percentile of the dSEVRaT obtained from label permuted data.

3 Results

We compared our proposed test, dSEVRaT, against other standard methods using both simulated data and real data. The simulated data is meant to show how dSEVRaT can outperform other methods without requiring 0-time normalization. The evaluation on real data shows the practical utility of dSEVRaT. We compared dSEVRaT against PEM and Durbin-Watson test (DW). The genes were ranked according to the statistic produced by each of these methods; ROC [16] (a plot of true positive rate vs false positive rate) and the corresponding AUC (Area Under ROC Curve) were calculated, higher the AUC better the performance. We have not shown the results of SAM-Slope, SAM-Area and EDGE since [19] has shown that PEM outperformed all these standard methods. Our experiments show that dSEVRaT outperforms PEM.

3.1 Simulated Data

We have carried out simulation experiment to understand the performance variation of dSEVRaT, PEM and DW for varying number of samples and varying amount of Signal-to-noise ratio. For this purpose, we have simulated a data of 10000 genes with 1000 (10%) being responsive and the remaining 9000 (90%) being unresponsive. The unresponsive genes' time-course is sum of unstimulated expression (m_i) and noise components (ϵ_{it}) i.e the function $\epsilon_{it} + m_i$. ϵ_{it} and m_i are sampled from normal distribution with 0 mean and variance σ^2. The responsive genes' time-course E_{it}s are sum of signal (S_t), unstimulated expression (m_i) and noise (ϵ_{it}) components i.e. the function $S_t + m_i + \epsilon_{it}$. S_t belongs to one of the four representative patterns of response as shown in figure 1: (1) linear; (2) step; (3) half sin; and, (4) quasi sin. We have simulated 250 genes with each of these patterns amounting to 1000 responsive genes in total. Linear pattern signifies gradual monotonous change of expression after stimulation; step pattern signifies sudden change of expression; half sin pattern signifies pulsive or peaked response and the quasi sin pattern signifies the change of early response to later response. We studied for $n \in \{7,12\}$ and S/N ratio $\in \{1,2\}$. After having simulated the data, each gene was 0-time normalized before applying the shown methods. The performance of various methods, in terms of AUCs, are summarized in table 1. It shows that dSEVRaT out performs both DW and PEM on all 0-time normalized data irrespective of n and S/N ratio. It further reveals that S/N ratio has higher impact on the performance than n.

3.2 Environmental Response Data

Yeast environmental stress response data was generated by Gasch et al [11] and Derisi et al [9] on nearly 6000 genes of yeast as a part of understanding the way

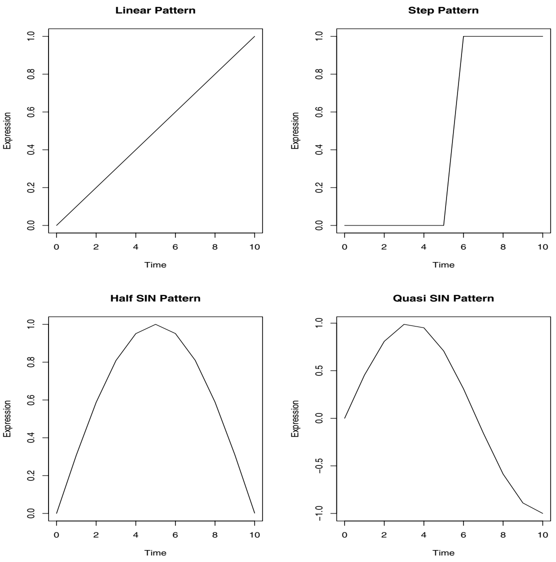


Fig. 1. Time-course gene expression response patterns used for simulatation studies

Table 1. Performance of dSEVRaT, PEM and DW in terms of AUC on simulated data. Each column shows performance on one data generated for one simulation parameters' setting. Relative performance of PEM is lower to that of DW for all parameter settings. But, dSEVRaT performs consistently better than both PEM and DW for all parameter settings. The Median column summarizes the median performance of all methods over all datasets; dSEVRaT is the best.

Method	n=7, S/N=1	n=7, S/N=2	n=12, S/N=1	n=12, S/N=2	Median
dSEVRaT	**0.635**	**0.844**	**0.659**	**0.877**	**0.752**
DW	0.619	0.787	0.657	0.846	0.722
PEM	0.577	0.720	0.604	0.768	0.662

yeast adopts or reacts to various stresses present in its environment. We have selected 10 datasets published by Gasch et al which contain minimum 7 time points, excluding 0-time point: (1) two nearly identical experiments (10 and 12 time points) on Stationary phase; (2) Menadione exposure with 9 time points; (3) Nitrogen source depletion with 10 time points; (4) Heat shock from 25^{o}C to 37^{o}C response sampled for 8 time points; (5) Hydrogen peroxide treatment consists of 10 time points; (6) Hyper-osmotic shock response sampled for 7 time points; (7) Diamide treatment response sampled at 8 time points; (8) Diauxic shift study, consisting of 7 time points; and (9) DTT exposure response, consisting of 8 time points; This dataset is a good benchmark to test our dSEVRaT against PEM and DW because of several datasets with varying number of sampling points and shapes of response from dataset to dataset and the availability of good

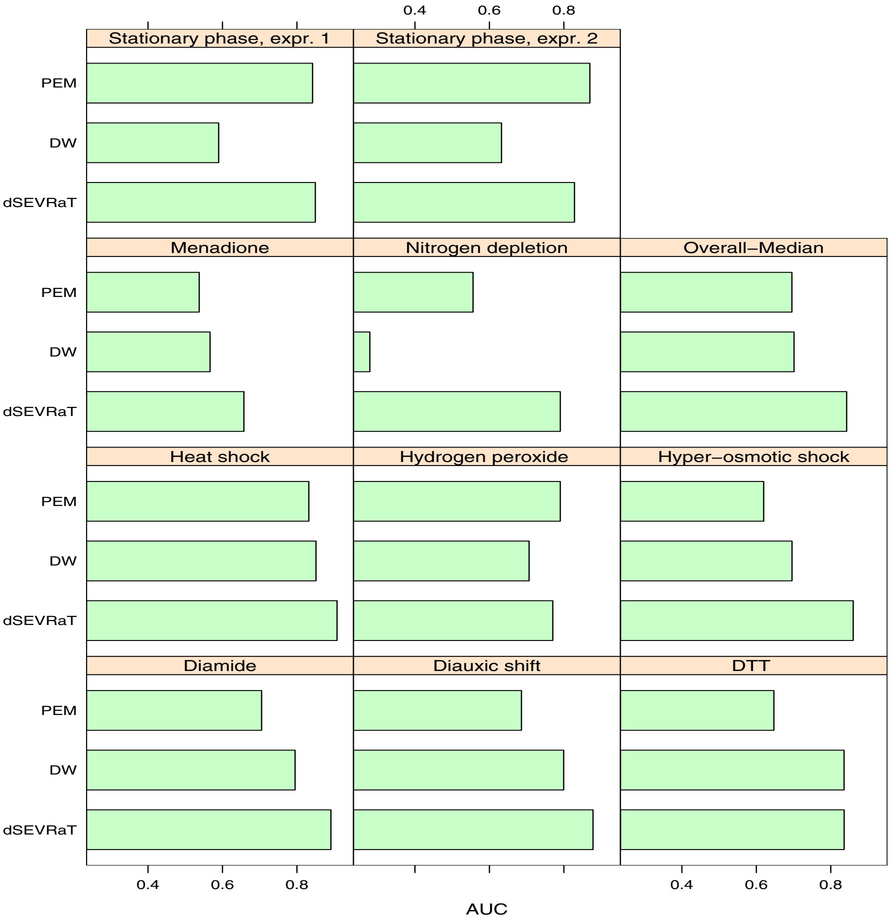


Fig. 2. Barcharts of performance of dSEVRaT, PEM and DW in terms of AUC on yeast environmental stress response data. Each panel shows performance on one stress data. Relative performance of PEM compared to DW changes from data to data. But, dSEVRaT performs consistently better than both PEM and DW on all datasets except on Stationary phase datasets. dSEVRaT is outperforming all methods on all datasets. The Overall-Median panel summarizes the median performance of all methods; dSEVRaT is the best again.

number of highly potential true positives. This dataset is a good example of 0-time normalized data.

dSEVRaT, PEM and DW were applied on these 10 datasets individually and their performance was evaluated based on a true positive set of 270 genes available at the website of Chen et al [8]. This is intersection of environmental stress response genes obtained by co-regulation study by Gasch et al [11] and the yeast orthologs of pombe stress response genes published by Chen et al. Barcharts in figure 2 show the performance of dSEVRaT and the other methods;

dSEVRaT outperforms both PEM and DW on all datasets except on stationary phase and hydrogen peroxide datasets. This shows the efficacy of the strategy used in the design of dSEVRaT. The slightly worse performance of dSEVRaT on these datasets is due to their step response with step taking place between 0th and 1st time-points.

3.3 Yeast Cell Cycle Data

The yeast cell cycle datasets used in the discovery of cell cycle regulated genes (Spellman et al. [22]) have been obtained by four cell synchronization experiments: (1) alpha factor, (2) CDC15, (3) CDC28 and (4) Elu. This is an example of non 0-time normalized data. We used the 104 genes used as true cell cycled regulated genes by Spellman et al as our true positive set the remaining as false positive set. In addition to PEM and DW, we also include Fourier Transform (FT) method for comparison [22][6] since FT is exclusively suitable for this task. Performance of dSEVRaT, PEM, DW and FT are shown in table 2. dSEVRaT performs better than DW and PEM on all datasets except on CDC15 dataset in which PEM does better. As expected, FT is consistently better than all these autocorrelation based methods. This is due to the generality of these autocorrelation methods and specialization of FT to identify cell cycle regulated genes. Yet, the performance of dSEVRaT is not far below that of FT and it performed better than DW and PEM.

Table 2. Performance of dSEVRaT, PEM and DW in terms of AUC on yeast cell cycle data from Spellman et al. Each column shows performance on one synchroniation data. PEM has consistently performed better than DW on all datasets. dSEVRaT performs better than PEM on all datasets except on CDC15. But, as expected, FT is the best method because FT especially looks for periodically expressed genes while dSEVRaT, PEM and DW look for smoothly varying genes which are not exclusively periodic. The Median column summarizes the median performance of all methods; dSEVRaT is the best among dSEVRaT, PEM and DW, it is not far below from FT.

Method	CDC28	Alpha	CDC15	Elu	Median
dSEVRaT	**0.793**	**0.879**	0.770	**0.772**	**0.783**
DW	0.703	0.826	0.763	0.749	0.756
PEM	0.747	0.862	**0.781**	0.761	0.771
FT	0.859	0.917	0.811	0.8	0.835

4 Discussion

We have shown the effect of 0-time normalization on the data and the consequent interpretation. From our analysis, it is clear that 0-time normalization introduces spurious level of average in the expression time-course of each gene, though its quantity may change from gene to gene. This is a dominant problem in the case of data with no repeats or insufficient repeats. To circumvent this problem, we proposed a test, called dSEVRaT, based on relative variance of the signal against its serial or first order autocorrelation. In this statistic, we enhance the variance factor in smoothness dependent manner which plays a major role in improving the performance without resorting to 0-time normalization. This helps us in avoiding the stray effects of 0-time normalization. Apart from 0-time normalization, this data-dependent enhancement of variance improves performance even in the cases where 0-time normalization is not required as in cell cycle data. We enhanced it by using two data dependent parameters: (1) minimum variance V_{min}; and, (2) fudge factor S_0.

We have carried out a simulation experiment to study the effect of number of samples (n) and the signal-to-noise ratio (S/N) on the performance of dSEVRaT, DW and PEM. It revealed that the performance improvement mainly comes from S/N improvement as compared to n. But, for all sets of parameter settings, we have shown that dSEVRaT outperformed DW as well as PEM. Whereas DW outperformed PEM owing to the fact that DW has not used 0-time normalization whereas PEM did. Our experiments on environmental stress response data has shown that dSEVRaT outperformed both DW and PEM tests where PEM outperformed the contemporarily popular methods such as SAM and EDGE both in terms of median performance as well as individual performance. The experiments on cell cycle data have shown that dSEVRaT outperformed all other methods except Fourier Transform method; this is acceptable because Fourier Transform method is the best suited method to identify cell cycle regulated genes. This performance of dSEVRaT is contributed mainly by smoothness enhancement and further improvement was made with fudge factor. dSEVRaT can also be applied even on data with repeats by simply averaging the repeats. Differentially responsive genes in case of multi-treatment aligned time-course data (i.e. one time-course for each treatment at comparable time-points) can be achieved either by taking the ratio of the two dSEVRaT statistics treatment and control/reference or applying dSEVRaT on the time-course obtained by taking the differences of treatment and control at each time-point.

The main limitation of dSEVRaT, autocorrelation based tests in general, is that it requires the sampling of expression has to be sufficient so that the smoothness of the undelying inherent response is reflected in the data. In other words, it requires a reasonable number of time points such as more than five.

Further, this method is limited by exploiting only the first order autocorrelation. We are currently working on generalized $dSEVRaT$ of order k, denoted as $dSEVRaT^k$, which exploits k-th order autocorrelation. Unlike in model error analysis [5], we have to combine $dSEVRaT^k$ of all orders appropriately to arrive at the final test since k may be different for different genes.

Acknowledgements

We thank Joshy George, Yudi Pawitan, Xu Han, Rehena Sultana, Vladamir Kuznetsov and Edison Liu for their valuable suggestions during this work. This research was supported by Genome Institute of Singapore.

References

1. Mukesh B, Giusy DG, and Diego dB: Inference of gene regulatory networks and compound mode of action from time course gene expression profiles, Bioinformatics, 22(7):815-822 (2006).
2. Benjamini Y and Hochberg Y: Controlling the false discovery rate: a practical and powerful approach to multiple testing. J. Roy. Stat. Soc. B. 1995; 57: 289-300.
3. Karuturi RKM and Vinsensius BV: Friendly NeighborsMethod for Unsupervised Determination of Gene Significance in Time-courseMicroarray Data, IEEE Symposium on Bioinformatics and Bioengineering, (2004).
4. Karuturi RKM, Silvia W, Wing-Kin S and Lance DM: Differential Friendly Neighbors Algorithm for Differential Relationships Based Gene Selection and Classification using Microarray Data, The 2007 Intl Conf on Data Mining (DMIN'06), (2006).
5. Durbin, J and Watson, GS: Testing for Serial Correlation in Least Squares Regression I & II, Biometrika, 37:409-428 (1950) & 38:159-179 (1951).
6. Karuturi RKM and Liu JH: Improved Fourier Transform Method for Unsupervised Cell-cycle Regulated Gene Prediction, IEEE Computational Systems Bioinformatics (2004).
7. Chao C, Xiaotu M, Xiting Y, Fengzhu S and Lei ML: MARD: a new method to detect differential gene expression in treatment-control time courses, Bioinformatics, 22(21):2650-2657 (2006).
8. Chen D, Toone WM, Mata J, Lyne R, Burns G, Kivinen K, Brazma A, Jones N, and Bhler J: Global transcriptional responses of fission yeast to environmental stress, Mol. Biol. Cell 2003; 14: 214- 229.
9. DeRisi JL, Iyer VR, and Brown PO: Exploring the metabolic and genetic control of gene expression on a genomic scale. Science 1997; 278: 680-686.
10. Eisen MB, Spellman PT, Brown PO, and Botstein D: Cluster analysis and display of genome-wide expression patterns. Proc. Natl Acad. Sci. USA 1998; 95: 14863-14868.
11. Gasch AP, Spellman PT, Kao CM, Carmel-Harel O, Eisen MB, Storz G, Botstein D, and Brown PO: Genomic expression programs in the response of yeast cells to environmental changes. Mol. Biol. Cell 2000; 11: 4241-4257.
12. Callegaro A, Basso D and Bicciato S: A locally adaptive statistical procedure (LAP) to identify differentially expressed chromosomal regions, Bioinformatics 22(21):2658-2666 (2006).
13. Efron B., Tibshirani R., Storey J.D. and Tusher V.: Empirical Bayes analysis of a microarray experiment, J. Am. Stat. Assoc., 96:11511160(2001).
14. Luan Y and Li H: Model-based methods for identifying periodically expressed genes based on time course microarray gene expression data, Bioinformatics, 20(3):332-339 (2004)
15. Ana C, Maria JN, Alberto F and Manuel T: maSigPro: a method to identify significantly differential expression profiles in time-course microarray experiments, Bioinformatics, 22(9):1096-1102 (2006).
16. McNeil BJ and Hanley JA: Statistical approaches to the analysis of receiver operating characteristic (ROC) curves. Med. Decis. Mak. 1984; 4: 137-150.
17. Park T, Yi S, Lee S, Lee SY, Yoo D, Ahn J, and Lee Y: Statistical tests for identifying differentially expressed genes in time-course microarray experiments. Bioinformatics 2003; 19: 694-703.

18. Peddada SD, Lobenhofer EK, Li L, Afshari CA, Weinberg CR, and Umbach DM: Gene selection and clustering for time-course and dose-response microarray experiments using order-restricted inference, Bioinformatics 2003; 19: 834-841.
19. Xu H, Wing-Kin S and Lin F: Pem: A General Statistical Approach for Identifying Differentially Expressed Genes in Time-course CDNA Microarray Experiment without Replicate, IEEE Symposium on Computational Systems Bioinformatics, (2006).
20. Xu P, Karuturi RKM, Lance DM, Kui L, Yonghui J, Pinar K, Long W, Lim-Soon W, Edison TL, Mohan KB and Jianhua L: Identification of Cell Cycle-regulated Genes in Fission Yeast, 16:1026-1042 (2005).
21. Smyth GK: Linear models and empirical bayes methods for assessing differential expression in microarray experiments. Statistical applications in genetics and molecular biology 2004; 3: article 3.
22. Spellman PT, Sherlock G, Zhang MO, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, and Futcher B: Comprehensive identification of cell-cycleregulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. Mol. Biol. Cell 1998; 9: 3273-3297.
23. Storey JD, Xiao W, Leek JT, Tompkins RG, and Davis RW: Significance analysis of time course microarray experiments. Proc. Natl Acad. Sci. USA, 102:12837-12842(2005).
24. Tusher V, Tibshirani R, and Chu G: Significance analysis of microarrays applied to the ionizing radiation response. Proc. Natl Acad. Sci. USA 98:5116-5121 (2001).
25. Xu XL, Olson JM, and Zhao LP: A regression-based method to identify differentially expressed genes in microarray time course studies and its application in an inducible Huntington.s disease transgenic model. Human Molecular Genetics, 1977-1985 (2002).
26. Jing T, Li Z, Xia J, Kevin KY, Karuturi RKM, and Qiang Y: Apoptosis Signal-regulating Kinase 1 Is a Direct Target of E2F1 and Contributes to Histone Deacetylase Inhibitorinduced Apoptosis through Positive Feedback Regulation of E2F1 Apoptotic Activity, Journal of Biological Chemistry, 281(15):10508-10515 (2006).

Efficiently Finding the Most Parsimonious Phylogenetic Tree Via Linear Programming

Srinath Sridhar[1], Fumei Lam[2], Guy E. Blelloch[1], R. Ravi[2],
and Russell Schwartz[3]

[1] Computer Science Department, Carnegie Mellon University
{guyb,srinath}@cs.cmu.edu
[2] Tepper School of Business, Carnegie Mellon University
lam@math.mit.edu, ravi@cmu.edu
[3] Department of Biological Sciences, Carnegie Mellon University
russells@andrew.cmu.edu

Abstract. Reconstruction of phylogenetic trees is a fundamental problem in computational biology. While excellent heuristic methods are available for many variants of this problem, new advances in phylogeny inference will be required if we are to be able to continue to make effective use of the rapidly growing stores of variation data now being gathered. In this paper, we introduce an integer linear programming formulation to find the most parsimonious phylogenetic tree from a set of binary variation data. The method uses a flow-based formulation that could use exponential numbers of variables and constraints in the worst case. The method has, however, proved extremely efficient in practice on datasets that are well beyond the reach of the available provably efficient methods. The program solves several large mtDNA and Y-chromosome instances within a few seconds, giving provably optimal results in times competitive with fast heuristics than cannot guarantee optimality.

1 Introduction

Phylogeny construction, or the inference of evolutionary trees from some form of population variation data, is one of the oldest and most intensively studied problems in computational biology, yet it remains far from solved. The problem has become particularly acute for the special case of intraspecies phylogenetics, or tokogenetics, in which we wish to build evolutionary trees among individuals in a single species. In part, the persistence of the problem reflects its basic computational difficulty. The problem in most reasonable variants is formally NP hard [15] and thus has no known efficient solution. The continuing relevance of phylogeny inference algorithms also stems from the fact that the data sets to be solved have been getting increasingly large in both population sizes and numbers of variations examined. The genomic era has led to the identification of vast numbers of variant sites for human populations [21,30], as well as various other complex eukaryotic organisms [29,11,10]. Large-scale resequencing efforts are now under way to use such sites to study population histories with precision

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 37–48, 2007.

never previously possible [9]. Even more vast data sets are available for microbial and viral genomes. As a result, methods that were adequate even a few years ago may no longer be suitable today.

In this work, we focus on the inference of intraspecies phylogenies on binary genetic variation data, which is of particular practical importance because of the large amount of binary SNP data now available. The binary intraspecies phylogeny problem has traditionally been modeled by the minimum Steiner tree problem on binary sequences, a classic NP hard problem [15]. Some special cases of the problem are efficiently solvable, most notably the case of *perfect phylogenies*, in which each variant site mutates only once within the optimal tree [1,16,22]. However, real data will not, in general, conform to the perfect phylogeny assumption. The standard in practice is the use of sophisticated heuristics that will always produce a tree but cannot guarantee optimality (e.g. [3,12,27]). Some theoretical advances have recently been made in the efficient solution of *near-perfect phylogenies*, those that deviate only by a fixed amount from the assumption of perfection [6,13,31,32]. These methods can provide provably efficient solutions in many instances, but still struggle with some moderate-size data sets in practice. As a result, some recent attention has turned to integer linear programming (ILP) methods [17]. ILPs provide provably optimal solutions and while they do not provide guaranteed run-time bounds, they may have practical run times far better than those of the provably efficient methods.

In the present work, we develop an ILP formulation to solve the most parsimonious phylogenetic tree problem on binary sequences. This method finds provably optimal trees from real binary sequence data, much like the prior theoretical methods and unlike the prevailing heuristic methods. Practical run time is, however, substantially lower than that of the existing provably efficient theoretical methods, allowing us to tackle larger and more difficult datasets. Below, we formalize the problem solved, present our methods, and establish their practical value on a selection of real variation data sets. These methods provide a platform for more extensive empirical studies of variation patterns on genomic scales than were previously possible. They may also help lay the groundwork for more sophisticated optimization methods that are likely to be needed in the future.

2 Preliminaries

We will assume that the input to the problem is a haplotype matrix H where each row corresponds to a haploid sequence of a taxon and each column corresponds to a binary marker such as a Single Nucleotide Polymorphism (SNP). The input H can therefore be viewed as an $n \times m$ binary matrix.

Definition 1. *A* phylogeny *T for input I is a tree where each vertex represents a binary string in $\{0,1\}^m$ and all the input sequences are represented in T. The* length *of T is the sum of the Hamming distances between all the adjacent vertices. The problem of constructing the most parsimonious (optimal) phylogeny is to find the phylogeny T^* such that* $\texttt{length}(T^*)$ *is minimized.*

Definition 2. *A phylogeny T for input I with m varying sites is q-near-perfect (or q-imperfect) if* $\texttt{length}(T) = m + q$.

The problem of reconstructing phylogenies is closely related to the *Steiner Tree Problem*, a well studied problem in combinatorial optimization (for a survey and applications, see [8,20]). Given a graph $G = (V, E)$ and a set of *terminals* in V, the problem is to find the smallest subgraph of G such that there is a path between any pair of terminals.

The problem can be related to the phylogeny construction problem as follows. Let graph G be the m-cube defined on vertices $V = \{0, 1\}^m$ and edges $E = \{(u, v) \in V \times V : \sum_i |u_i - v_i| = 1\}$. The vertices are binary strings of length m and an edge connects two vertices if and only if their Hamming distance is 1. Let $V_T \subseteq V$ be the set of species corresponding to the rows of input matrix H. The maximum parsimony problem is then equivalent to the minimum Steiner tree problem on underlying graph G with terminal vertices V_T. Even in this restricted setting, the Steiner tree problem has been shown to be NP-complete [14]. However, the phylogeny reconstruction problem when the optimal phylogeny is q-near-perfect can be solved in time polynomial in n and m when $q = O(\log(\text{poly}(n, m)))$ [32]. If q is very large, though, such algorithms do not perform well. Moreover, these algorithms use a sub-routine that solves the Steiner tree problem on m-cubes when the dimensions are small. Therefore, improving the existing solutions for the general problem will also improve the running time for the restricted cases.

3 Preprocessing

We now describe a set of preprocessing steps that can substantially reduce the size of the input data without affecting the final output.

3.1 Reducing the Set of Possible Steiner Vertices

The complexity of solving the Steiner tree problem in general graphs is a consequence of the exponentially many possible subsets that can be chosen as the final set of Steiner vertices in the most parsimonious phylogeny. Therefore, an important component of any computational solution to the Steiner tree problem is to eliminate vertices that cannot be present in any optimal tree. We describe an approach that has been used to eliminate such vertices when the underlying graph is the m-cube. For input matrix H and column c of H, the *split* $c(0)|c(1)$ defined by c is a partition of the taxa into two sets, where $c(0)$ is the set of taxa with value 0 in column c and $c(1)$ is the set of taxa with value 1 in column c. This forms a partition of the taxa since $c(0) \cup c(1)$ is the set of all taxa and $c(0) \cap c(1)$ is empty. Each of $c(0)$ and $c(1)$ is called a *block* of c. Buneman used the blocks of binary taxa to introduce a graph, now called the *Buneman graph* $\mathcal{B}(H)$, which captures structural properties of the optimal phylogeny [7]. We will explain the generalization of this graph due to Barthélemy [4,28]. Each vertex of the Buneman graph is an m-tuple of blocks $[c_1(i_1), c_2(i_2), \ldots c_m(i_m)]$ ($i_j = 0$ or

```
function findBuneman($V_T$)

  1. let $\lambda \leftarrow V_T$; let $v \in \lambda$
  2. bunemanNeighbor($\lambda, v$)

function bunemanNeighbor($\lambda, v$)

  1. for all $j \in \{1, \ldots, m\}$
     (a) let $v' \leftarrow v; v'_j \leftarrow c_j(1 - i_j)$
     (b) if $v'$ is Buneman and $v' \notin \lambda$ then
           i. $\lambda \leftarrow \lambda \cup \{v'\}$
          ii. bunemanNeighbor($\lambda, v'$)
```

Fig. 1. Finding the Buneman graph in polynomial time

1 for each $1 \leq j \leq m$), with one block for each column and such that each pair of blocks has nonempty intersection ($c_j(i_j) \cap c_k(i_k) \neq \emptyset$ for all $1 \leq j, k \leq m$). There is an edge between two vertices in $\mathcal{B}(H)$ if and only if they differ in exactly one block. Buneman graphs are very useful because of the following theorem.

Theorem 1. *[3,28] For input matrix H, let T^*_H denote the optimal phylogeny on H and let $\mathcal{B}(H)$ denote the Buneman graph on H. If matrix H has binary values, then every optimal phylogeny T^*_H is a subgraph of $\mathcal{B}(H)$.*

Using the above theorem, our problem is now reduced to constructing the Buneman graph on input H and solving our problem on underlying graph $\mathcal{B}(H)$. Ideally we would like to find the Buneman graph in time $O(\text{poly}(k))$ where k is the number of vertices in the Buneman graph. Note that this is output-sensitive. We first state the following theorem, which we will use to show the Buneman graph can be generated efficiently.

Theorem 2. *[28] The Buneman graph $\mathcal{B}(H)$ is connected for any input matrix H in which all columns contain both states $0, 1$ and all pairs of columns are distinct.*

To generate the graph $\mathcal{B}(H)$, let $i_1, i_2, \ldots i_m$ be the first taxon in H. Then $v = [c_1(i_1), c_2(i_2), \ldots c_m(i_m)]$ is a vertex of $\mathcal{B}(H)$. Now, there are several ways to generate the graph $\mathcal{B}(H)$. The pseudo-code in Figure 1 begins with V_T the set of vertices of the $\mathcal{B}(H)$ corresponding to H. The algorithm then iteratively selects a vertex v and enumerates all the neighbors. For each vertex, the algorithm checks if it obeys the conditions of the Buneman graph, if so it is added to λ and we recurse.

Lemma 1. *The algorithm in Figure 1 finds the Buneman graph $\mathcal{B}(H)$ for the given input in time $O(km)$ where k is the number of vertices in $\mathcal{B}(H)$.*

Proof. The algorithm begins with a vertex $v \in \mathcal{B}(H)$ and determines $\mathcal{B}(H)$ in the depth-first search order. By Theorem 2, the algorithm will visit all vertices in $\mathcal{B}(H)$. Step 1a iterates over all m possible neighbors of vertex v in the m-cube which takes time $O(m)$. For each vertex $v \in \mathcal{B}(H)$ function `bunemanNeighbor` is called using v exactly once. Therefore if there are k vertices in $\mathcal{B}(H)$, then the time spent to discover all of $\mathcal{B}(H)$ is $O(km)$. Note that instead of using depth-first search, we could use breadth-first search or any other traversal order. □

3.2 Decomposition into Smaller Problems

In addition to allowing us to reduce the set of possible Steiner vertices, we show how Theorem 1 also allows us to decompose the problem into independent subproblems.

Definition 3. *[2] A pair of columns i, j* conflict *if the matrix H restricted to i, j contains all four gametes $(0,0), (0,1), (1,0)$ and $(1,1)$. Equivalently, the columns* conflict *if the projection of H onto dimensions i, j contains all four points of the square.*

For input I, the structure of the conflicts of I provides important information for building optimal phylogenies for I. For example, it is well known that a perfect phylogeny exists if and only if no pair of columns conflict [16,28]. In order to represent the conflicts of H, we construct the *conflict graph* $\mathcal{G}$, where the vertices of $\mathcal{G}$ are columns of H and the edges of $\mathcal{G}$ correspond to pairs of conflicting columns [18]. The following theorem has been stated previously without proof [18]. For the sake of completeness, we provide an explicit proof in the full paper using Theorem 1 and ideas from Gusfield and Bansal [18]. We denote the matrix H restricted to set of columns C as $C(H)$.

Theorem 3. *Let χ denote the set of non-trivial connected components of conflict graph $\mathcal{G}$ and let V_{isol} denote the set of isolated vertices of $\mathcal{G}$. Then any optimal Steiner tree on H is a union of optimal Steiner trees on the separate components of $\mathcal{G}$ and $\texttt{length}(T^*_H) = |V_{isol}| + \sum_{C \in \chi} \texttt{length}(T^*_{C(H)})$.*

Our decomposition preprocessing step proceeds as follows. We first construct the conflict graph $\mathcal{G}$ for input matrix H and identify the set of connected components of $\mathcal{G}$. We ignore the columns corresponding to the isolated vertices V_{isol} since they each contribute exactly one edge to the final phylogeny. Then the columns corresponding to each connected component c of χ can be used independently to solve for the most parsimonious phylogeny. Our problem is now reduced to input matrices H consisting of a single non-trivial connected component.

3.3 Merging Rows and Columns

We now transform the input matrix H to possibly reduce its size. We can remove rows of H until all the rows are distinct since this does not change the phylogeny. Furthermore, we can remove all the columns of H that do not contain both states 0 and 1 since such columns will not affect the size or the topology of the phylogeny. Finally, we will assign weights w_i to column i; w_i is initialized to

1 for all i. We iteratively perform the following operation: identify columns i and j that are identical (up to relabeling 0, 1), set $w_i := w_i + w_j$ and remove column j from the matrix. Notice that in the final matrix H, all pair-wise rows are distinct, all pair-wise columns are distinct (even after relabeling 0, 1), every column contains both 0, 1 and all the columns have weights $w_i \geq 1$. From now, the input to the problem consists of the matrix H along with vector w containing the weights for the columns of H. We can now redefine the length of a phylogeny using a weighted Hamming distance as follows.

Definition 4. *The* `length` *of phylogeny $T(V, E)$ is* $\texttt{length}(T) = \sum_{(u,v)\in E} \sum_{i\in D(u,v)} w_i$*, where $D(u, v)$ is the set of indices where u, v differ.*

It is straight-forward to prove the correctness of the pre-processing step.

Lemma 2. *The length of the optimal phylogeny on the pre-processed input is the same as that of the original input.*

4 ILP Formulation

A common approach for studying the minimum Steiner tree problem is to use integer and linear programming methods. For convenience, we will consider the more general problem of finding a minimum Steiner tree for directed weighted graphs G (we represent an undirected graph as a directed graph by replacing each edge by two directed edges). The input to the minimum directed Steiner tree problem is a directed graph, a set of terminals T and a specified root vertex $r \in T$. The minimum Steiner tree is the minimum cost subgraph containing a directed path from r to every other terminal in T.

For a subgraph S of graph G, we associate a vector $x^S \in \mathbb{R}^E$, where edge variable x^S_e takes value 1 if e appears in the subgraph S and 0 otherwise. A subset of vertices $U \subset V$ is *proper* if it is nonempty and does not contain all vertices. For $U \subset V$, let $\delta^+(U)$ denote the set of edges (u, v) with $u \in U$, $v \notin U$ and for a subset of edges $F \subseteq E$, let $x(F) = \sum_{e\in F} x_e$. Finally, edge-weights are given by $w_e \in R^E$.

The problem of finding a minimum directed Steiner tree rooted at r has previously been examined with an ILP based on graph cuts [5,24,35]:

$$\min \textstyle\sum_{u,v} w_{u,v}x_{u,v} \tag{1}$$

$$\text{subject to } x(\delta^+(U)) \geq 1 \ \forall \text{ proper } U \subset V \text{ with } r \in U,\, T \cap \overline{U} \neq \emptyset \tag{2}$$

$$x_{u,v} \in \{0, 1\} \ \text{ for all } (u, v) \in E. \tag{3}$$

Constraints (2) impose that r has a directed path to all terminal vertices T. Note that in our phylogenetic tree reconstruction problem, the underlying graph for the problem is the Buneman graph and any input taxon can be chosen as the root vertex r. Since the Buneman graph may have an exponential number of vertices and edges with respect to the size of the input matrix H, the running time for solving this integer program may be doubly-exponential in m in the worst case.

We develop an alternative formulation based on multicommodity flows [35]. In this formulation, one unit of flow is sent from the root vertex to every terminal vertex. Every terminal vertex except the root acts as a sink for one unit of flow and the Steiner vertices have perfect flow conservation. We use two types of binary variables $f^t_{u,v}$ and $s_{u,v}$ for each edge $(u, v) \in E$. The variables $f^t_{u,v}$ are real valued and represent the amount of flow along edge (u, v) whose destination is terminal t. Variables $s_{u,v}$ are binary variables denoting the presence or absence of edge (u, v). The program is then the following:

$$\begin{aligned}
&\min && \textstyle\sum_{u,v} w_{u,v} s_{u,v} && && (4)\\
&\text{subject to} && \textstyle\sum_v f^t_{u,v} = \sum_v f^t_{v,u} && \text{for all } u \notin T, t \in T && (5)\\
& && \textstyle\sum_v f^t_{v,t} = 1, \sum_v f^t_{t,v} = 0, \sum_v f^t_{r,v} = 1 && \text{for all } t \in T && (6)\\
& && 0 \le f^t_{u,v} \le s_{u,v} && \text{for all } t \in T && (7)\\
& && s_{u,v} \in \{0, 1\} && \text{for all } (u, v) \in E. && (8)
\end{aligned}$$

Constraints (5) impose the condition of flow conservation on the Steiner vertices. Constraints (6) impose the inflow/outflow constraints on terminals in T. Finally, constraints (7) impose the condition that there is positive flow on an edge only if the edge is selected. By the max-flow min-cut theorem, the projection of the solution onto the variables s satisfy constraints (2) [24]. The results will thus satisfy the following theorem:

Theorem 4. *All integer variables of the above linear program are binary and the solution to the ILP gives a most parsimonious phylogenetic tree.*

5 Empirical Results

We applied the ILP to several sets of variation data chosen to span a range of data characteristics and computational difficulties. We used only non-recombining data (Y chromosome, mtDNA, and bacterial DNA) because imperfection in non-recombining data is most likely to be explained by recurrent mutations. We used two Y chromosome data sets: a set of all human Y chromosome data from the HapMap [21] and a set of predominantly chimpanzee primate data [33]. Several different samples of mitochondrial DNA(mtDNA) were also included [34,26,23,19]. Finally, we analyzed a single bacterial sample [25].

The pre-processing and ILP formulation was performed in C++ and solved using the Concert callable library of CPLEX 10.0. In each case, the ILP was able to find an optimal tree on the data after preprocessing. We also used the `pars` program of `phylip` which attempts to heuristically find the most parsimonious phylogeny. `pars` was run with default parameters. Empirical tests were conducted on a 2.4 GHz Pentium 4 computer with 1G RAM, running Linux. We attempted to use the `penny` program of `phylip`, which finds provably optimal solution by branch-and-bound, but it terminated in under 20 minutes only for the smallest mitochondrial data set and was aborted by us after 20 minutes for all other tests.

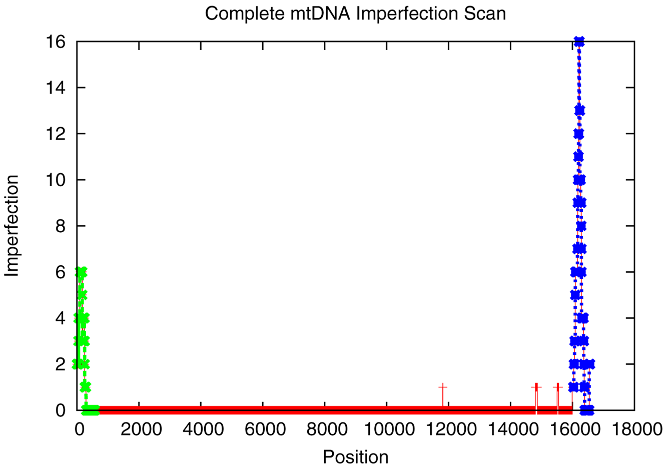


Fig. 2. Imperfection of the most parsimonious phylogeny for overlapping windows across the complete mitochondrial genome. The x-axis shows the sites in their order along the genomic axis. The y-axis shows the imperfection for the window centered on the corresponding site. The hyper variable D-loop region (1 . . . 577 and 16028 . . . 16569) shows significantly larger imperfection.

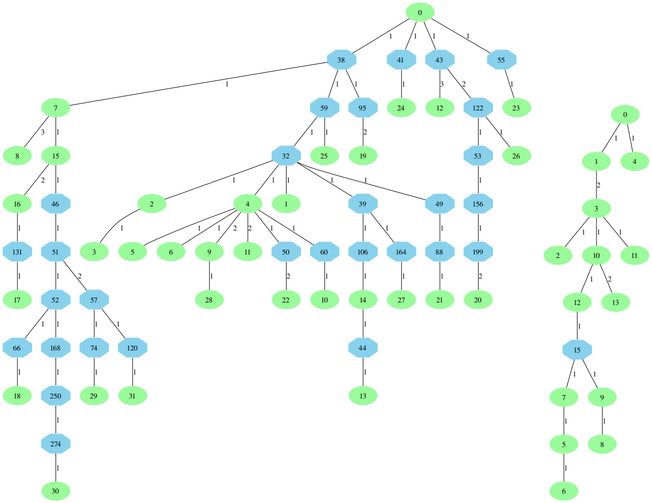


Fig. 3. Examples of trees of varying levels of difficulty. (a) Human mitochondrial data from Wirth et al. [34] (b) Human Y chromosome from HapMap [21].

We first used the mitochondrial data as a basic validation of the correctness of the methods and the reasonableness of the maximum parsimony criterion on these data. The HVS-I and HVS-II segments of the mitochondrial D-loop region

Table 1. Empirical run-time results on a selection of non-recombining datasets

	input			time(secs)	
Data Set	before	after	length	our ILP	`pars`
human chrY [21]	150 × 49	14 × 15	16	0.02	2.55
bacterial [25]	17 × 1510	12 × 89	96	0.08	0.06
mtDNA chimp [33]	24 × 1041	19 × 61	63	0.08	2.63
y chimp [33]	15 × 98	15 × 98	99	0.02	0.03
human mtDNA [34]	40 × 52	32 × 52	73	13.39	11.24
human mtDNA [19]	395 × 830	34 × 39	53	53.4	712.95
human mtDNA [26]	13 × 390	13 × 42	48	0.02	0.41
human mtDNA [23]	33 × 405	27 × 39	43	0.09	0.59

have exceptionally high mutation rates [34], providing a good test case of the ability of our algorithm to distinguish regions we would expect to have perfect or near-perfect phylogenies from those expected to have highly imperfect phylogenies. Figure 2 shows a scan of 201-site long windows across the complete 16569-site mtDNA genome. Since the mtDNA is circular, the windows wrap around over the ends in the genome order. The y-axis corresponds to *imperfection*, which is the number of recurrent mutations in the most parsimonious phylogeny. The figure does indeed show substantially larger phylogenies within the high mutation rate D-loop region ($1 \ldots 577$ and $16028 \ldots 16569$) than in the low mutation rate coding regions, confirming the relevance of a parsimony metric for such data sets.

We then ran the methods on a collection of data sets to assess efficiency of the methods. Figure 3 provides two examples of most parsimonious phylogenies for data sets at opposite extremes of difficulty: an mtDNA sample [34] with imperfection 21 (Fig. 3(a)) and the human Y chromosome sample, with imperfection 1 (Fig. 3(b)). Table 1 presents the empirical run-time data for all of the datasets. The columns 'input before' and 'input after' correspond to the size of the original input and that after preprocessing. Run times vary over several orders of magnitude and appear largely insensitive to the actual sizes of the data sets. Rather, the major determinant of run time appears to be a dataset's imperfection, i.e., the difference between the optimal length and the number of variant sites. It has recently been shown that the phylogeny problem under various assumptions is fixed parameter tractable in imperfection [6,13,31,32] possibly suggesting why it is a critical factor in run time determination. The `pars` program of `phylip`, despite providing no guarantees of optimality, does indeed find optimal phylogenies in all of the above instances. It is, however, slower than the ILP in most of these cases.

6 Conclusion

We have developed an ILP formulation for optimally solving for the most parsimonious phylogeny using binary genome variation data. The method fills an important practical need for fast methods for generating provably optimal trees from large SNP variation datasets. This need is not served well by the heuristic methods that are currently the standard for tree-building, which generally work well in practice but cannot provide guarantees of optimality. More recent theoretical methods that find provably optimal trees within defined run-time bounds are inefficient in practice without a fast sub-routine to solve the general problem on smaller instances. The ILP approach allows extremely fast solutions of the easy cases while still yielding run-times competitive with a widely used fast heuristic for hard instances. Methods such as ours are likely to be increasingly important as data sets accumulate on larger population groups and larger numbers of variant sites.

Acknowledgments

We thank Daniel Gusfield for helpful discussions and for motivating our use of LP for problems in phylogenetics. This work was supported by U.S. National Science Foundation grants IIS-0612099, CCR-0105548, and CCR-0122581 (The ALADDIN project).

References

1. R. Agarwala and D. Fernandez-Baca. A polynomial-time algorithm for the perfect phylogeny problem when the number of character states is fixed. *SIAM Journal on Computing*, 23:1216–1224, 1994.
2. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10:323–340, 2003.
3. H. J. Bandelt, P. Forster, B. C. Sykes, and M. B. Richards. Mitochondrial portraits of human populations using median networks. *Genetics*, 141:743–753, 1989.
4. J. Barthélemy. From copair hypergraphs to median graphs with latent vertices. *Discrete Math*, 76:9–28, 1989.
5. J. E. Beasley. An algorithm for the Steiner problem in graphs. *Networks*, 14:147–159, 1984.
6. G. E. Blelloch, K. Dhamdhere, E. Halperin, R. Ravi, R. Schwartz, and S. Sridhar. Fixed parameter tractability of binary near-perfect phylogenetic tree reconstruction. *International Colloquium on Automata, Languages and Programming*, 2006.
7. P. Buneman. The recovery of trees from measures of dissimilarity. *Mathematics in the Archeological and Historical Sciences, F. Hodson et al., Eds.*, pages 387–395, 1971.
8. X. Cheng and D. Z. Du (Eds.). *Steiner Trees in Industry.* Springer, 2002.
9. The ENCODE Project Consortium. The ENCODE (ENCyclopedia Of DNA Elements) Project. *Science*, 306(5696):636–640, 2004.
10. Lindblad-Toh et al. Genome sequence, comparative analysis and haplotype structure of the domestic dog. *Nature*, 438(7069):803–819, 2005.
11. Lindblad-Toh K et al. Large-scale discovery and genotyping of single-nucleotide polymorphisms in the mouse. *Nature Genetics*, pages 381–386, 2000.
12. J. Felsenstein. PHYLIP (Phylogeny Inference Package) version 3.6. distributed by the author, Department of Genome Sciences, University of Washington, Seattle, 2005.

13. D. Fernandez-Baca and J. Lagergren. A polynomial-time algorithm for near-perfect phylogeny. *SIAM Journal on Computing*, 32:1115–1127, 2003.
14. L. R. Foulds and R. L. Graham. The Steiner problem in phylogeny is NP-complete. *Advances in Applied Mathematics*, 3, 1982.
15. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
16. D. Gusfield. Efficient algorithms for inferring evolutionary trees. *Networks*, 21:19–28, 1991.
17. D. Gusfield. Haplotyping by pure parsimony. *Combinatorial Pattern Matching*, 2003.
18. D. Gusfield and V. Bansal. A fundamental decomposition theory for phylogenetic networks and incompatible characters. *Research in Computational Molecular Biology*, 2005.
19. A. Helgason, G. Palsson, H. S. Pedersen, E. Angulalik, E. D. Gunnarsdottir, B. Yngvadottir, and K. Stefansson. mtDNA variation in Inuit populations of Greenland and Canada: migration history and population structure. *American Journal of Physical Anthropology*, 130:123–134, 2006.
20. F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*, volume 53. Annals of Discrete Mathematics, 1992.
21. The International HapMap Consortium. The International HapMap Project. `www.hapmap.org`. *Nature*, 426:789–796, 2005.
22. S. Kannan and T. Warnow. A fast algorithm for the computation and enumeration of perfect phylogenies. *SIAM Journal on Computing*, 26:1749–1763, 1997.
23. C. M. Jr. Lewis, R.Y. Tito, B. Lizarraga, and A. C Stone. Land, language, and loci: mtDNA in Native Americans and the genetic history of Peru. *American Journal of Physical Anthropology*, 127:351–360, 2005.
24. N. Maculan. The Steiner problem in graphs. *Annals of Discrete Mathematics*, 31:185–212, 1987.
25. M. Merimaa, M. Liivak, E. Heinaru, J. Truu, and A. Heinaru. Functional coadaption of phenol hydroxylase and catechol 2,3-dioxygenase genes in bacteria possessing different phenol and p-cresol degradation pathways. *Eighth Symposium on Bacterial Genetics and Ecology*, 31:185–212, 2005.
26. Sharma S, Saha A, Rai E, Bhat A, and Bamezai R. Human mtDNA hypervariable regions, HVR I and II, hint at deep common maternal founder and subsequent maternal gene flow in Indian population groups. *American Journal of Human Genetics*, 50:497–506, 2005.
27. N. Saitou and M. Nei. The neighbor-joining method: a new method for reconstructing phylogenetic trees. *Molecular Biology and Evolution*, 4(4):406–425, 1987.
28. C. Semple and M. Steel. *Phylogenetics*. Oxford University Press, 2003.
29. The Chimpanzee Sequencing and Analysis Consortium. Initial sequence of the chimpanzee genome and comparison with the human genome. *Nature*, 437(7055):69–87, 2005.
30. Elizabeth M. Smigielski, Karl Sirotkin, Minghong Ward, and Stephen T. Sherry. dbSNP: a database of single nucleotide polymorphisms. *Nucleic Acids Research*, 28(1):352–355, 2000.
31. S. Sridhar, G. E. Blelloch, R. Ravi, and R. Schwartz. Optimal imperfect phylogeny reconstruction and haplotyping. *Computational Systems Bioinformatics*, 2006.
32. S. Sridhar, K. Dhamdhere, G. E. Blelloch, E. Halperin, R. Ravi, and R. Schwartz. Simple reconstruction of binary near-perfect phylogenetic trees. *International Workshop on Bioinformatics Research and Applications*, 2006.

33. A. C. Stone, R. C. Griffiths, S. L. Zegura, and M. F. Hammer. High levels of Y-chromosome nucleotide diversity in the genus Pan. *Proceedings of the National Academy of Sciences USA*, pages 43–48, 2002.
34. Thierry Wirth, Xiaoyan Wang, Bodo Linz, Richard P. Novick, J. Koji Lum, Martin Blaser, Giovanna Morelli, Daniel Falush, and Mark Achtman. Distinguishing human ethnic groups by means of sequences from Helicobacter pylori: Lessons from Ladakh. *Proceedings of the National Academy of Sciences USA*, 101(14):4746–4751, 2004.
35. R. T. Wong. A dual ascent approach for Steiner tree problems on a directed graph. *Mathematical Programming*, 28:271–287, 1984.

A Multi-Stack Based Phylogenetic Tree Building Method

Róbert Busa-Fekete[1], András Kocsor[1], and Csaba Bagyinka[2]

[1] Research Group on Artificial Intelligence of the Hungarian Academy of Sciences
and University of Szeged, H-6720 Szeged, Aradi vértanúk tere 1., Hungary
{busarobi,kocsor}@inf.u-szeged.hu
[2] Institute of Biophysics, Biological Research Center of the Hung. Acad. Sci.
H-6701 Szeged, P. O. 521., Hungary
csaba@nucleus.szbk.u-szeged.hu

Abstract. Here we introduce a new Multi-Stack (MS) based phylogenetic tree building method. The Multi-Stack approach organizes the candidate subtrees (i.e. those having same number of leaves) into limited priority queues, always selecting the K-best subtrees, according to their distance estimation error. Using the K-best subtrees our method iteratively applies a novel subtree joining strategy to generate candidate higher level subtrees from the existing low-level ones. This new MS method uses the Constrained Least Squares Criteria (CLSC) which guarantees the non-negativity of the edge weights.

The method was evaluated on real-life datasets as well as on artificial data. Our empirical study consists of three very different biological domains, and the artificial tests were carried out by applying a proper model population generator which evolves the sequences according to the predetermined branching pattern of a randomly generated model tree. The MS method was compared with the Unweighted Pair Group Method (UPGMA), Neighbor-Joining (NJ), Maximum Likelihood (ML) and Fitch-Margoliash (FM) methods in terms of Branch Score Distance (BSD) and Distance Estimation Error (DEE). The results show clearly that the MS method can achieve improvements in building phylogenetic trees.

Keywords: Phylogenetics – tree estimation – Multi-Stack – tree-joining operator.

1 Introduction

The reliable reconstruction of a tree topology from a set of homologous sequence data is one of the most important goals in system biology. A major family of the phylogenetic tree building methods is the *distance-based* or *distance matrix methods*. The general idea behind them is to calculate a measure for the distance between each pair of taxa, and then find a tree that predicts the observed set of distances as closely as possible. There are quite a few heuristic distance-based algorithms with a fixed criterion available for estimating phylogeny, and their strengths and weaknesses are familiar to everyone in the field. The distance-based

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 49–60, 2007.

methods, like the Unweighted Pair-Group Method using Arithmetic averages (UPGMA) [1] and Neighbor-Joining (NJ) [2], work similarly: they iteratively form clusters, always choosing the best possibility based on a given criterion. We can call these methods greedy in a certain sense, because they always work on the current best candidate subtrees. The NJ method produces additive trees, while UPGMA assumes that the evolutionary process can be represented by an ultrametric tree. These restrictions may then interfere with the correct estimation of the evolutionary process.

The chief aim of this paper is to develop a good distance-based method that closely approximates to the true tree for any available evolutionary (not just for ultrametric or additive) distance. To achieve this we apply a special form of the Least Square Criteria (LSC) to phylogenetic trees [4]. The LSC will guarantee a minimal deviation between the evolutionary distances and the leaf distances in the phylogenetic tree. It is fortunate that the LSC weighting for a phylogenetic tree can be computed in $O\left(n^2\right)$ time. The original LSC was introduced by Fitch and Margoliash, and nowadays several forms of it are in use in the literature, like the Weighted LSC [5], Unweighted and Generalized LSC [6]. We applied the constrained version of LSC (CLSC) here to evaluate phylogenetic trees because the weights of the edges have to be non-negative. The solution of the problem retains its simplicity because the Constrained LSC can easily be handled by the Levenberg-Marquardt method [7].

Since finding the least squares tree (whether it is constrained or not) is an NP-complete problem [8], a polynomial-time algorithm to solve it is unlikely to exist. Many meta-heuristics have been applied in phylogenetic tree-building. We now propose a novel heuristic, based on the so-called Multi-Stack (MS) construction [10]. The MS heuristic organizes the candidate subtrees having the same number of leaves into a priority queue according to their distance estimation error, and generates newer candidate trees by joining the existing trees via a novel tree joining strategy. It may happen however that there are many trees within a priority queue that have a non-disjunct set of leaves, and it is not possible to join them. The Closest-Neighbourhood Tree Joining (CNTJ) strategy introduced here always provides a tree topology based on all of the subtrees, swapping their common taxa with their closest neighbour. Our method was tested on artificial as well as on real-life datasets.

2 Background

2.1 Phylogenetic Trees

A tree is a connected acyclic graph. First we denote the vertex set and the edge set of a tree T by $V\left(T\right)$ and $E\left(T\right)$, respectively. Furthermore, let us denote the non-negative weights of the edges by $w : E\left(T\right) \rightarrow \mathbb{R}_{\geq 0}$. A weighted tree assigns a distance for each pair of leaves (which can be calculated by summing the weight of the edges on the path between them) that is called the *leaf* distance of T, and will be denoted by D^T. A *phylogenetic tree* is represented as a leaf-labelled weighted binary tree. The labels of the leaves of phylogenetic tree T correspond to the set of taxa $\mathcal{T}_T$. The inner nodes represent the hypothetical ancestors, and the weighting of the phylogenetic tree represents the evolutionary distance defined by T. If we regard T as a rooted tree, then there is only one internal node of degree 2; the degrees of the other internal nodes are 3. Here we will only deal with rooted phylogenetic trees. The subset of the descendants of an internal node is called a cluster, and the internal nodes are the most recent *common ancestors* of the *monophyletic group* or *cluster*. Thus the internal nodes of a phylogenetic tree and clusters are equivalent concepts. This way each phylogenetic tree corresponds to a set of compatible clusters $\mathcal{C}$ (i.e. for all $A, B \in \mathcal{C}$ either $A \subseteq B$, or $B \subseteq A$, or $A \cap B = \emptyset$). This construction is also called a *Linnean Hierarchy.* Here we will denote the clusters of T by $T^{\mathcal{C}}$.

The Robinson-Foulds (RF) distance or *symmetric difference* for rooted trees [11] is based on this approach as well. Because the RF distance of two rooted phylogenetic trees T_1 and T_2 is the cardinality of the symmetric difference of their cluster sets, $T_1^{\mathcal{C}} \triangle T_2^{\mathcal{C}} = \left(T_1^{\mathcal{C}} \setminus T_2^{\mathcal{C}}\right) \cup \left(T_2^{\mathcal{C}} \setminus T_1^{\mathcal{C}}\right)$. There is also an extension of the RF distance introduced by Kuhner and Felsenstein [12], and it is known as the Branch Score Distance (BSD).

2.2 Constrained Least Squares Criterion

There are many criteria in use for phylogenetic trees that can be applied to the distance data, like Minimum Evolution Length and Least Squares (LSC) Criterion. There are many forms of Least Squares Criteria available, and all of them require the optimization of a quadratic function. Before we formally describe these criteria, let us denote the *path-edge incidence* or *topology matrix* of a phylogenetic tree T by P_T. The matrix P_T is a binary matrix whose columns correspond to the edges of T, while the rows correspond to the paths between the leaves of T. This representation of a tree requires a space of $O\left(n^3\right)$ because it has $n-1$ columns and $\binom{n}{2}$ rows, even though it has just a few non-zero elements. Hence it is worth exploiting the sparsity of the topology matrix for an efficient implementation.

Next, we will denote a distance matrix by D defined on the taxon set of $\mathcal{T}$. We can rewrite D using its vector form $\mathbf{d}$ (i.e. turning the upper triangular of D into a vector). The arrangement according to the topology matrix P_T determines an unambiguous ordering among the $\binom{n}{2}$ entries of the vector $\mathbf{d}$. Introducing the necessary notations, we can write, in a simple way, the Unweighted Least Square Criteria (LSC) for a given T phylogenetic tree. The edge weighting of a tree T satisfies the LSC criteria if it satisfies the following optimisation task:

$$\min_{\mathbf{x} \in \mathbb{R}^{\mathbf{n-1}}} \|\left(P_T \mathbf{x} - \mathbf{d}\right)\| \tag{1}$$

where the elements of $\mathbf{x}$ may have any real value, including zero or negative values. The solution of the problem defined by Eq. (1) results in an optimal edge weighting for a phylogenetic tree T and a minimal Frobenius norm for $\|D_T - D\|_F$. This means that the deviation between the calculated weighting D_T and D is minimal. The problem can be solved in $O\left(n^2\right)$ time for a given phylogenetic tree [5]. We also require that a weighting always be positive because a negative evolutionary distance has no physical sense. That is why we will restrict ourselves here to the following minimization problem:

$$\min_{\mathbf{x}\in\mathbb{R}^{\mathbf{n-1}}} \| \left(P_T\mathbf{x} - \mathbf{d}\right) \| \qquad s.t. \quad \mathbf{0} \leq \mathbf{x} \tag{2}$$

The Constrained Least Squares Criteria (CLSC) defined above results in a non-negative weighting for a phylogenetic tree T. CLSC also retains the property of the original LSC that it can be solved in $O\left(n^2\right)$ time, because the algorithm introduced by Bryant & Waddell [5] can handle the Levenberg-Marquardt method as well. Here $\|D_T - D\|_F$ is the *distance estimation error (DEE)*, and

$$\frac{\|D_T - D\|_F}{\binom{n}{2}}$$

is the *normalized distance estimation error (NDEE)* of the phylogenetic tree and will be denoted by e_T.

3 Materials and Methods

3.1 Multi-Stack Approach

To solve problems which have enormous solution spaces we need to apply an efficient search technique. That is why we decided to adopt a heuristic approach for phylogenetic tree-building which is also used in speech recognition [13]. To describe the method we first have to give a definition. A *stack* is a structure for keeping candidate solutions in. Furthermore, we use limited-sized stacks: if there are too many candidates in a stack, we prune the ones with the highest fitness value. In the MS algorithm we assign a separate stack to the trees having the same number of leaves and store the K-best candidate subtrees in the stack according to their DEEs. In the initial step the algorithm generates the lower level subtrees only, and then it pops each pair of candidate subtree from the stacks, joins them in every possible way, and afterwards puts the new candidate subtrees into the stack according their leaf numbers associated with their new DEE. Applying this heuristic to phylogenetic tree building, we obtain an iterative tree-building procedure.

The pseudocode of the MS algorithm is presented in Table 1, where the Q_n elements denote the limited priority queue which contains at most K trees, and each tree has exactly n leaves in it. The initial step of the method includes the exploration of all tree topologies with at most three leaves. This step makes sense because there are only $n + \binom{n}{2} + \binom{n}{3}$ phylogenetic trees when $|\mathcal{T}| = n$, so during this initial step we can explore the whole space of trees. In the next steps MS generates the possible subtrees, and it always keeps the best K subtrees based on their distance estimation error.

The complexity of the MS method naturally depends on the variable K because we are exploring an n^2K^2 tree topology. Since CLSC requires a quadratic

Table 1. The Multi-Stack algorithm

```
Input:  D distance matrix, $K$ size of priority queues
1       Initial step: fill up $Q_1, Q_2$ and $Q_3$
2       for $i = 3 : n$
3           for $j = 1 : min(n - i, i)$
5               Generate all of the trees joining the
                    elements of $Q_i$ and $Q_j$
6               Add them to the
                    priority queue $Q_{i+j}$ according their DEE
7           endfor
8       endfor
9       return to first element of $Q_n^K$
Output: Rooted phylogenetic tree $T$ with $n$ leaves
```

time complexity $(O\left(n^2\right))$, it becomes the most time-consuming step of the MS. Due to this features the MS tree building method has a time complexity of $O\left(K^2 n^4\right)$ overall. This computation also includes the time requirements of the joining step which will be introduced in the next section.

3.2 Closest Neighborhood Tree Joining Operator

With the Multi-Stack tree building approach it may happen that we want to join two candidate trees that have some common taxa. The simplest idea is the naive approach: let us replace the common taxon set of the candidate trees that interferes the tree joining in every possible way with those taxa that do not occur in the taxon set of candidate trees. After we have carried out and evaluated all possible replacements, let us choose the best replacement. But it can be easily seen that this will lead to a very high computational burden, because the number of possible replacement grows exponentially with the number of the common taxon set. Instead here we suggest a tree joining strategy as a way of avoiding this problem.

We need to join two candidate subtrees T_1, T_2 having n_1 and n_2 leaves respectively, and we need to determine a strategy for the elimination of the duplicated taxa of the candidate trees: $|\mathcal{T}_{T_1} \cap \mathcal{T}_{T_2}| = k$. From the solution of this problem we also require that the distance estimation errors e_{T_1} and e_{T_2} with respect to the applied distance matrix D remain or grow as little as possible. Thus the goal here is to determine a strategy for the replacement of common taxa that produce the least variation in the tree estimation errors of the candidate trees in question.

For the formal description let us denote the cost of the replacement for a taxon $t \in \mathcal{T}_T$ by $c(t, t^{'})$, where $t^{'} \in \mathcal{T} - \mathcal{T}_T$. This leads to a change in the e_T value after the replacement, which can be a negative real number as well. We can readily determine an upper bound for this cost, because using the weights of T before the replacement, the following proposition always hold.

Proposition 1. *Let T be a phylogenetic tree with a taxon set $\mathcal{T}_T \subset \mathcal{T}$, and let e_T be its distance estimation error. A distance on $\mathcal{T}$ will be denoted by D,* *and the leaf distance will be denoted by D^T. Now let $t \in \mathcal{T}_T$ and $t^{'} \in \mathcal{T} - \mathcal{T}_T$ be two taxons. Then the following inequality will hold for the $c(t, t^{'})$ cost of the replacement:*

$$c(t, t^{'}) \leq \sum_{t^{''} \in \mathcal{T}_T} b_{t^{''}}(t, t) - b_{t^{''}}(t^{'}, t) \tag{3}$$

where $b^1_{t^{''}}(t_1, t_2) = \left| D(t_1, t^{''}) - D^T(t_2, t^{''}) \right|$

Proof. If we use the CLSC for T, then we get an optimal edge weighting w using the taxon set $\mathcal{T}_T$ and distance matrix D. Equation 3 corresponds to the rows in Equation 2, that is, it is represents the path between t and $\mathcal{T} - t$. Thus if we replace this taxon, the change of the optima will vary according to the magnitude when we use the weighting w. So Equation 3 will hold apart from the choice of $t^{'} \in \mathcal{T}$.

Summarizing the above points, Proposition 1 allows us to determine an upper bound for a replacement of a taxon. That is why we suggest here that the common taxon set $|\mathcal{T}_{T_1} \cap \mathcal{T}_{T_2}|$ should be replaced iteratively, taxon by taxon, always choosing the pair of taxons that have the lowest bound in accordance with Proposition 1.

3.3 Distances and Similarities

Evolutionary distances. The global alignment of protein sequences can be performed using the well-known Needlemann-Wunsch [14] algorithm with the BLOSUM70 [15] matrix. The simplest evolutionary distance between a pair of aligned sequences is usually measured by the number of sites where a substitution occurs. Many models have been proposed to describe the true evolutionary process. There are many corrections of this measure which try to fine tune the evolutionary rate. Some of them were used here when we performed our tests on different real-life datasets. These include the Gamma, Poisson and Jukes-Cantor corrections [16].

Compression-based similarity measures. The *information theoretical distance* functions are based on a comparison of how many information sequences there are relative to each other. This approach originated from Kolmogorov-complexity theory. The Conditional Kolmogorov complexity $K(X|Y)$ is defined as the length of the shortest program computing X on an input Y [19]. The Kolmogorov complexity $K(X)$ of a sequence X is a shorthand notation for $K(X|\lambda)$, where λ is an empty sequence. The corresponding distance function uses the relative decrease in complexity or conditional complexity as a measure of sequence similarity, that is

$$d(X, Y) = \frac{max\{K(X|Y), K(Y|X)\}}{K(YX)} \tag{4}$$

Kolmogorov complexity is a non-computable notion, so in practical applications it is approximated by the length of a compressed sequence calculated with a compression algorithms like LZW [20] or Sequitur [21]. The formula for calculating compression-based similarity measures (CBM) using the length values of compressed sequences can be derived from Equation 4. It takes the form

$$d_{CBM}\left(X,Y\right)=\frac{C\left(XY\right)-min\{C\left(X\right),C\left(Y\right)\}}{max\{C\left(X\right),C\left(Y\right)\}} \tag{5}$$

where $C\left(.\right)$ denotes the length of a compressed sequence, compressed by a particular compressor C. We will focus on two well-known compressor algorithm in our experiments, namely LZW [20] and Sequitur [21].

3.4 Generation of Model Populations

Since the correct phylogeny for a set of taxa is usually unknown, we first carried out our tests on randomly generated model populations having $10-20-30-40$ members. For each population 100 independent and identically-distributed and and non ultrametric model trees were generated from the tree-space. In order to calculate the leaves of these trees, pseudo random sequences of 600 amino acids were used as ancestor sequences. The sequence was then assumed to evolve according to the predetermined branching pattern of the randomly generated model tree. The edge lengths of the generated tree correspond to the expected number of amino acid substitutions per site. We varied this value between $0-0.1$, and the number of amino acid substitutions at each site was assumed to have a Poisson distribution [17,18], were also used to mimic the mutations. Using this rate we carried out point mutations according to the BLOSUM70 matrix. Hundred different set of sequences (model populations) were generated for each (10-20-30-40) member number.

3.5 Description of Real-Life Datasets

We utilized three different datasets of various size to compare and test the methods. Primates consist of mitochondrial DNA, while hydrogenases and myoglobins are distinct protein families, hence they are very suitable objects for statistically testing different tree building and distance (similarity) calculating procedures.

The set of *primates* is quite small (12 sequences), and it was borrowed from Ovchinnikov et al. [22]. This dataset contains the mitochondrial DNA of two Neanderthals, the modern human species and other vertebrates. The second set we used for testing is a typical set of sequences of *myoglobins*. It contains 27 proteins from different organisms. The third set is the group of 75 [NiFe] *hydrogenases*. Hydrogenases are metalloenzymes that catalyze the reaction $H_2 \rightleftharpoons 2H^+ + 2e^-$. They can be found in bacteria, archae and cyanobacteria. The [NiFe] hydrogenases are usually placed into 4 different taxonomic groups [23]. In the rest of the paper these datasets will be called *primates, myoglobins and hydrogenases* respectively.

4 Experiments

4.1 Evaluation of the Model Populations

The evolutionary distances (Poisson distance and CBM similarity measure with the Sequitur compressor method [21]) were calculated for the model populations, and phylogenetic trees were built over these model populations using four different tree-building methods: Unweighted Pair Group Method with Arithmetic Mean (UPGMA) [1], Neighbour-Joining (NJ) [2], Maximum Likelihood method for proteins (ML) [24] and the Fitch-Margoliash (FM) [4] method, all of which were implemented in the Phylip package [25], and our newly developed Multi-Stack method (MS). The parameter K for the MS method was set to 20 for the populations having 10 and 20 members (leaves) and to 40 for the populations with 30 and 40 members (leaves).

The BSD distance between the randomly-generated model tree and the built phylogenetic tree along with the distance estimation error (DEE) were calculated after building the phylogenetic tree. The test was repeated 100 times on 100 similar model populations and the average of BSD distance and DEE was calculated. The results of this are summarized in Table 2. It is striking that the MS method is superior to all other methods tested. Both the BSD and the DEE values are smaller in every case when the MS method was applied. The UPGMA approach in contrast proved to be the least efficient method in reconstructing phylogenetic trees. The performances of the NJ and the FM method are quite similar, and the means of the BSD distance are equal to each other in many test cases. The mean of the Normalized DEE and BSD distances for NJ and FM only lags behind the results of the MS method by a small amount when the leaf number is set to 40.

Table 2. The performance of the test on randomly generated model trees. The values in bold show the minimal value in each row.

	No	Length of ancestor = 600				
	leaves	UPGMA	NJ	FM	MS	ML
Poisson-Poisson DEE(*10^3)	10	60.83	24.69	25.13	**7.39** ($K = 20$)	55.6
	20	41.10	16.62	16.58	**10.33** ($K = 20$)	37.5
	30	30.45	11.85	11.97	**6.85** ($K = 40$)	29.9
	40	24.86	9.37	9.35	**5.12** ($K = 40$)	21.6
Poisson-Sequitur DEE(*10^3)	10	122.55	34.79	35.90	**17.49** ($K = 20$)	191.4
	20	70.58	16.58	16.32	**11.05** ($K = 20$)	101.3
	30	48.45	10.31	10.39	**6.61** ($K = 40$)	67.8
	40	37.32	6.16	6.06	**5.50** ($K = 40$)	69.1
Poisson-Poisson BSD distance	10	0.32	0.21	0.22	**0.20** ($K = 20$)	0.28
	20	0.50	0.33	0.34	**0.28** ($K = 20$)	0.33
	30	0.63	0.41	0.41	**0.32** ($K = 40$)	0.56
	40	0.73	0.48	0.48	**0.38** ($K = 40$)	0.62
Poisson-Sequitur BSD distance	10	0.49	0.27	0.29	**0.29** ($K = 20$)	0.34
	20	0.79	0.44	0.44	**0.32** ($K = 20$)	0.39
	30	0.95	0.54	0.54	**0.35** ($K = 40$)	0.59
	40	1.15	0.63	0.62	**0.39** ($K = 40$)	0.62

4.2 Real-Life Datasets

The newly developed MS method was tested on real-life datasets as well. To evaluate the trees we used the distance estimation error values (DEE). The properties of the MS method were investigated and the results were again compared with other tree building algorithms. In this case we applied them on six evolutionary distances (similarities) (Jukes-Cantor, Gamma, Poisson, LZW, Sequitur and alignment score).

The only tunable parameter for the MS method is K (the size of the limited priority queue). When evaluating our MS trees we always chose the best tree in the last stack, i.e. the one which had the lowest DEE value. It is interesting to see the "goodness" of different trees in the last stack i.e. how much the "best tree" was better than the others. We set the value of parameter K to 30 and plotted the DEE value of the trees in the last stack (Figure 1) for primates, myoglobins and hydrogenases. The DEE for the trees grew slightly at the beginning of the stack, but the first few trees were almost as good. There was a pronounced jump after this nearly constant level. The position of the jump depends on the evolutionary distance used, but correlates with the number of leaves on the tree when the number of leaves is small (< 30). For hydrogenases the jump was around $K = 30$. In order to investigate the effect of the K parameter on the "goodness" of trees we also built trees for each dataset using different K values. The DEE value of the best tree (which has the smallest DEE) in the last queue was plotted against the K in Figure 2 for different phylogenetic distances and datasets. As can be seen, the DEE decreased while the limits of the priority queues rose to 60. Moreover there is threshold (about $30 - 40$), after which the DEE of the best trees remains practically constant.

As a rule of thumb these points give us a good estimation of what the parameter setting for K should be. According to this rule, K should be around the number of leaves if it is smaller than 30. For bigger trees $K = 40$ seems to be a good estimate. Applying this rule we built trees with different tree building methods using various distances on the three real-life datasets. The results are summarized in Table 3. It is evident that DEE in most cases is the smallest for our new MS based tree building method. The performance of the UPGMA was not as good as the others when compared with the model populations, but the

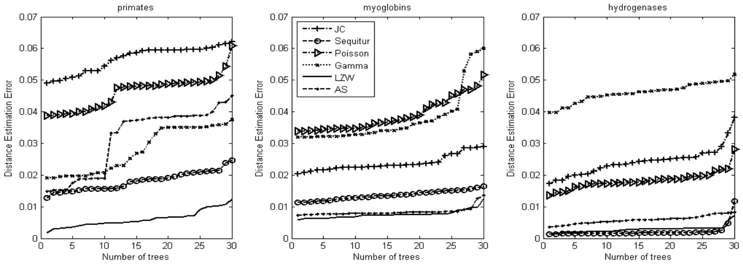


Fig. 1. The normalized DEE of the MS trees in the last priority queue ($K = 30$)

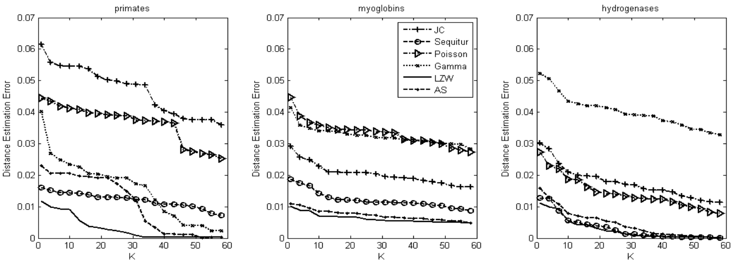


Fig. 2. The dependence of the normalized DEE of the best tree in the priority queue on the parameter K

Table 3. The normalized distance estimation error of different tree building methods using distinct similarity measures on the datasets. The values in bold show the minimal value in each row. Normalized distance estimation errors were multiplied by 1000. The value K for MS was set to 30 for primates and Myoglobins and 40 for hydrogenases. In this table UP means the UPGMA method, and A-S the Alignment-Score.

	Primates ($N = 12$)				Myoglobins ($N = 27$)				Hydrogenases ($N = 75$)			
	UP	NJ	FM	MS	UP	NJ	FM	MS	UP	NJ	FM	MS
JC	96.22	60.42	49.93	**45.17**	88.35	62.77	62.12	**19.70**	40.80	11.69	**10.58**	15.17
Gamma	112.87	76.36	63.08	**17.62**	174.56	90.69	81.43	**30.80**	56.44	18.29	**16.97**	37.48
Poisson	93.85	53.99	47.97	**37.06**	115.63	57.34	59.53	**32.58**	37.86	10.33	**8.76**	12.36
LZW	68.95	12.31	12.31	**1.68**	33.10	8.63	62.97	**5.60**	15.33	2.91	1.85	**0.50**
Sequ.	30.49	30.49	30.49	**11.89**	69.20	23.10	48.76	**10.99**	22.86	2.13	2.14	**0.52**
A-S	88.25	78.32	65.59	**13.71**	65.66	18.21	48.05	**7.03**	26.71	4.07	5.32	**1.27**

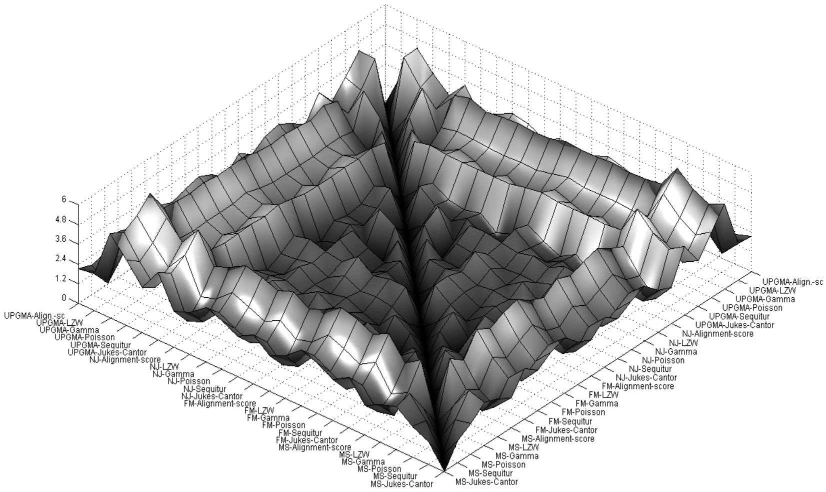


Fig. 3. The BSD distance of the trees with the myoglobin dataset

Normalized DEE for the FM and NJ methods are very similar here. Interestingly these two methods (NJ, FM) outperform the MS method in terms of a Normalized DEE when we used alignment-based evolutionary distances. Otherwise the MS method achieved better results. We also compared the methods in terms of their BSD values. In Figure 3 the labels along the axis represent the tree building methods and the evolutionary distance/similarity measure we applied and are separated by a hyphen. Comparing the tree topologies of different trees that employ the BSD, we see that the performance of the MS method is very similar for all datasets (note the plateau in Figure 3). It is also apparent from the evaluations that distance-based methods (UPGMA, NJ and FM) produce trees with very similar topologies (note the wide valley in the middle of the diagrams). The topology of the trees built by the MS method are different for these trees (notice the higher regions of the diagram in Figure 3). The MS method, however, produced similar topologies regardless of the evolution distance used for tree building, but we can still say that the MS trees brought an improvement in the Normalized DEE for the trees as Table 3 quite clearly indicates.

5 Conclusion

In this paper we have presented a novel distance-based and iterative tree building algorithm for analysing the lineage of taxa in structural biology and then compared it with other tree building methods using a new phylogenetic benchmark. Next we showed for simulated, model datasets and for three distinct real-life datasets that it is an efficient tool for building phylogenetic trees. The new method is superior on distance estimation and produces robust trees as the tests on model trees shows. The "goodness" of the resultant trees, however, strongly depends on the parameter K. As it follows from the nature of the method, if K is big enough the MS method approximates the exhaustive search. On choosing a proper K value, MS successfully and quickly searches in a previously unexplored region of the possible tree topologies, hence it produces slightly different topologies than those with the algorithms used previously. This allows us to gain a deeper insight into protein and DNA evolution, relationships and lineage, and we hope that the MS method and the phylogenetic benchmarking we introduced here will become widely used tools for tackling phylogenetic problems.

References

1. Rohlf F. J. (1963) Classification of Aedes by numerical taxonomic methods (Diptera: Culicidae). Ann Entomol Soc Am 56:798804.
2. Saitou N., Nei M. (1987) The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol Biol Evol. Jul;4(4):406-25.
3. Atteson K. (1999): The performance of neighbor-joining methods of phylogenetic reconstruction. Algorithmica, 25.
4. Fitch, W. M., and E. Margoliash. (1967): Construction of phylogenetic trees. Science 155:279284.

5. Bryant D. and Waddell P. (1998): Rapid Evaluation of Least-Squares and Minimum-Evolution Criteria on Phylogenetic Trees. Mol. Biol. Evol. 15(10):1346-1359.
6. Cavalli-Sforza, L., and Edwards. A. (1967): Phylogenetic analysis models and estimation procedures. Evolution 32: 550570.
7. Levenberg-Marquardt nonlinear least squares algorithms in C/C++ http://www.ics.forth.gr/~lourakis/levmar/
8. Day, W.H.E. (1986): Computational complexity of inferring phylogenies from dissimilarity matrices. Bulletin of Mathematical Biology 49:461-467.
9. Goloboff, P., A. (1999): Analysing large data sets in reasonable times: Solutions for composite optima. Cladistics 15:415-428, 1999.
10. Bahl L.R., Gopalakrishnan P.S. and Mercer R.L. (1993): Search Issues in Large Vocabulary Speech Recognition, Proceedings of the 1993 IEEE Workshop on Automatic Speech Recognition, Snowbird, UT.
11. Robinson, D.F., Foulds, L.R. (1981): Comparison of phylogenetic trees. Math. Biosci. 53, 131–147.
12. Kuhner M. K., Felsenstein J. (1995): A simulation comparison of phylogeny algorithms under equal and unequal evolutionary rates. Mol Biol Evol May;12(3):525.
13. Gosztolya G., Kocsor A. (2003): Improving the Multi-stack Decoding Algorithm in a Segment-Based Speech Recognizer. IEA/AIE 2003: 744-749
14. Needleman, S. B., Wunsch, C. D. (1970): A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48:443-453.
15. Henikoff S., Henikoff JG. (1992): Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. USA. 15;89(22):10915-9.
16. Jukes T. H., Cantor C. R. (1969): Evolution of protein molecules, pp. 21132 in Mammalian Protein Metabolism, edited by H. N. MUNRO. Academic Press, New York.
17. Zuckerland, E., and L. Pauling, (1965): Molecular disease, evolution, and genetic heterogeneity, pp. 189225 in Horizons in Biochemistry, edited by M. KASHA and B. PULLMAN. Academic Press, New York.
18. Dickerson, R. E. (1971): The structures of cytochrome c and the rates of molecular evolution. J. Mol. Evol. 1:26-45.
19. Cilibrasi R., Vitányi P. (2004): Clustering by compression, IEEE Transactions on Infomation Theory.
20. Ziv J., Lempel A. (1977): A universal algorithm for sequential data compression, IEEE Trans. on Inf. Th. IT-23 337-343.
21. Nevill-Manning C. G., Witten I. H. (1997): Compression and explanation using hierarchical grammars. Computer Journal, 40(2/3):103-116.
22. Ovchinnikov I., et al. (2000). Molecular analysis of Neanderthal DNA from the northern Caucasus, Nature 404(6777):490-493.
23. Vignais P. M., Billoud B., Meyer J. (2001): Classification and phylogeny of hydrogenases. FEMS Microbiology Reviews 25 455-501.
24. Felsenstein, J. (1981): Evolutionary trees from DNA sequences: a maximum likelihood approach. J. Mol. Evol. 17:368376.
25. Phylip program package http://evolution.genetics.washington.edu

A New Linear-Time Heuristic Algorithm for Computing the Parsimony Score of Phylogenetic Networks: Theoretical Bounds and Empirical Performance*

Guohua Jin[1], Luay Nakhleh[1], Sagi Snir[2], and Tamir Tuller[3]

[1] Department of Computer Science, Rice University, Houston, TX 77005, USA
{jin,nakhleh}@cs.rice.edu
[2] Department of Mathematics, University of California, Berkeley, CA 94720, USA
ssagi@math.berkeley.edu
[3] School of Computer Science, Tel Aviv University, Tel Aviv, Israel
tamirtul@post.tau.ac.il

Abstract. Phylogenies play a major role in representing the interrelationships among biological entities. Many methods for reconstructing and studying such phylogenies have been proposed, almost all of which assume that the underlying history of a given set of species can be represented by a binary tree. Although many biological processes can be effectively modeled and summarized in this fashion, others cannot: recombination, hybrid speciation, and horizontal gene transfer result in *networks*, rather than trees, of relationships.

In a series of papers, we have extended the maximum parsimony (MP) criterion to phylogenetic networks, demonstrated its appropriateness, and established the intractability of the problem of scoring the parsimony of a phylogenetic network. In this work we show the hardness of approximation for the general case of the problem, devise a very fast (linear-time) heuristic algorithm for it, and implement it on simulated as well as biological data.

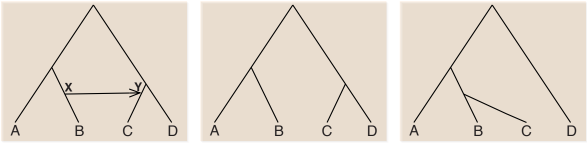
1 Introduction

Phylogenetic networks are a special class of *directed acyclic graphs* (DAGs) that models evolutionary histories when trees are inappropriate, such as in the cases of horizontal gene transfer (HGT) and hybrid speciation [26, 30, 27]. Fig. 1(a) illustrates a phylogenetic network on four species with a single HGT event. In horizontal gene transfer (HGT), genetic material is transferred from one lineage to another, as in Fig. 1(a). In an evolutionary scenario involving horizontal transfer, certain sites (specified by a specific substring within the DNA sequence of the species into which the horizontally transferred DNA was inserted) are inherited through horizontal transfer from another species (as in Figure 1(c)), while all others are inherited from the parent (as in Figure 1(b)). Thus, *each site evolves down one of the trees induced by (or, contained in) the network*. Similar scenarios arise in the cases of other reticulate evolution events (such as hybrid speciation and interspecific recombination).

* The authors appear in alphabetical order.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 61–72, 2007.

HGT plays a major role in bacterial genome diversification (e.g., see [7, 8, 19, 20]), and is a significant mechanism by which bacteria develop resistance to antibiotics (e.g., see [9]). Therefore, in order to reconstruct and analyze evolutionary histories of these groups of species, as well as to reconstruct the prokaryotic branch of the Tree of Life, developing accurate criteria for reconstructing and evaluating phylogenetic networks and efficient algorithms for inference based on these criteria is imperative. A large number of publications have been introduced in recent years about various aspects of phylogenetic networks; e.g., see [12, 30, 32, 11, 17, 18, 1, 31] for a sample of such papers in the last two years, and [26, 27] for detailed surveys.



(a) (b) (c)

Fig. 1. (a) A phylogenetic network with a single HGT event from X to Y. (b) The underlying organismal (species) tree. (c) The tree of a horizontally transferred gene.

In this work, we consider the *maximum parsimony* (MP) criterion, which has been in wide use for phylogenetic tree inference and evaluation. Roughly speaking, inference based on this criterion seeks the tree that minimizes the amount of evolution (in terms of number of mutations). In 1990, Jotun Hein proposed using this criterion for inferring the evolution of sequences subject to recombination. Recently, Nakhleh *et. al.* formulated the parsimony criterion for evaluating and inferring general phylogenetic networks [31], and we have recently demonstrated its appropriateness on both simulated and biological datasets [21, 22]. Applying the parsimony criterion for phylogenetic networks involves solving the *big* and the *small* parsimony problems, referred to as the **FTMPPN** and **PSPN** problems, respectively, in [31]. In [21] the small problem (scoring the parsimony of a given network) was proved to be NP-hard and a heuristic algorithm was devised. A recent work by Nguyen *et. al.* [33] provided a hardness result for a related, yet different, version of the small parsimony problem.

In this paper we devise a very fast (linear-time) heuristic algorithm, with very good empirical performance, for the PSPN problem. Further, we show that for a restricted, yet realistic, class of phylogenetic networks, our algorithm gives a polynomial time 3-approximation for the problem. Moreover, we show that although the theoretical approximation ratio is not very promising, the algorithm does give very good results in practice compared to the exact algorithm.

2 Parsimony of Phylogenetic Networks

Preliminaries and Definitions Let $T = (V, E)$ be a tree, where V and E are the *tree nodes* and *tree edges*, respectively, and let $\mathcal{L}(T)$ denote its leaf set. Further, let $\mathcal{X}$ be a set of taxa (species). Then, T is a phylogenetic tree over $\mathcal{X}$ if there is a bijection between $\mathcal{X}$ and $\mathcal{L}(T)$. Henceforth, we will identify the taxa set with the leaves they are mapped to, and let $[n] = \{1, .., n\}$ denote the set of leaf-labels. A tree T is said to be *rooted* if the set of edges E is directed and there is a single distinguished internal vertex r with in-degree 0. We denote by T_v the subtree rooted at v induced by the tree edges. A function $\lambda : [n] \rightarrow \{0, 1, .., \Sigma - 1\}$ is called a *state assignment function* over the alphabet Σ for T. We say that function $\hat{\lambda} : V(T) \rightarrow \{0, 1, .., \Sigma - 1\}$ is an extension of λ on T if it agrees with λ on the leaves of T. In a similar way, we define a function $\lambda^k : [n] \longmapsto \{0, 1, .., \Sigma - 1\}^k$ (in applications of the methodology, k corresponds to the sequence length) and an extension $\hat{\lambda}^k : V(T) \longmapsto \{0, 1, .., \Sigma - 1\}^k$. The latter function is called a *labeling* of T. We write $\hat{\lambda}^k(v) = s$ to denote that sequence s is the label of the vertex v. The ith *site* is an n-tuple where the jth coordinate is the state of the ith site of species (leaf) j.

Given a labeling $\hat{\lambda}^k$, let $d_e(\hat{\lambda}^k)$ denote the Hamming distance between the two sequences labeling the two endpoints of the edge $e \in E(T)$.

A phylogenetic network $N = N(T) = (V', E')$ over the taxa set $\mathcal{X}$ is derived from $T = (V, E)$ by adding a set H of edges to T, where each edge $h \in H$ is added as follows: (1) split an edge $e \in E$ by adding new node, v_e; (2) split an edge $e' \in E$ by adding new node, $v_{e'}$; (3) finally, add a directed *reticulation edge* from v_e to $v_{e'}$. It is important to note that the resulting network must be acyclic [30].

We extend the notion of T_v to networks as follows. For a network N and a node $v \in V(N)$, let N_v be the graph induced by all the nodes reachable from v. Finally, we denote by $\mathcal{T}(N)$ the set of all trees contained inside network N. Each such tree is obtained by the following two steps: (1) for each node of in-degree 2, remove one of the incoming edges, and then (2) for every node x of in-degree and out-degree 1, whose parent is u and child is v, remove node x and its two adjacent edges, and add a new edge from u to v.

Further, Phylogenetic networks must satisfy additional temporal constraints [30]. First, N should be acyclic (genetic material flows only forward in time). Second, N should satisfy additional temporal constraints, so as to reflect the biological fact that the donor and recipient of a horizontally transferred gene must co-exist in time. Since at the scale of evolution HGT events are instantaneous in time, a reticulation edge between two points dictates that they correspond to the same chronological time. This in turn implies that if x and y are the two endpoints of an HGT edge and their time-stamp is t, then there cannot be an HGT edge between a node z at time $t' < t$ and a node w at time $t'' > t$. Note that this condition is not guaranteed by the acyclicity condition[1]. See [30] for a formal description of the temporal constraints on phylogenetic networks.

2.1 Parsimony of Phylogenetic Networks

We begin by reviewing the parsimony criterion for phylogenetic trees.

[1] It is important to note that, while acyclicity must be satisfied by all phylogenetic networks, the other temporal constraints may be violated, due to extinction or incomplete taxon sampling, for example.

Problem 1. Parsimony Score of Phylogenetic Trees (PSPT)

> **Input:** A 3-tuple (S, T, λ^k), where T is a phylogenetic tree and λ^k is the labeling of $\mathcal{L}(T)$ by the sequences in S.
> **Output:** The extension $\hat{\lambda}^k$ that minimizes the expression $\sum_{e \in E(T)} d_e(\hat{\lambda}^k)$.

We define the parsimony score for (S, T, λ^k), $pars(S, T, \lambda^k)$, as the value of this sum, and $pars(S, T, \lambda^k, i)$ as the value of this sum for site i only. In other words, $pars(S, T, \lambda^k) = \sum_{1 \leq i \leq k} pars(S, T, \lambda^k, i)$. It is easy to see that the optimal value is obtained by optimal solutions for every site $1 \leq i \leq k$. Problem 1 has a polynomial time dynamic programming type algorithm originally devised by Fitch [10] and later extended by Sankoff [36]. The algorithm finds an optimal assignment (i.e., $\hat{\lambda}^k$) for each site separately.

Since Fitch's algorithm is a basic building block in this paper, we hereby describe it. As mentioned above, the input to the problem is a tree T and a single character $C = \lambda^1$. The algorithm finds the optimal assignment to internal nodes of T, in two phases: (1) assigning values to internal nodes in a bottom-up fashion, and (2) eliminating the values determined in the previous phase in a top-down fashion. Specifically, phase (1) proceeds as follows: for a node v with children v_1 and v_2 whose values $A(v_1)$ and $A(v_2)$ have been determined,

$$A(v) = \begin{cases} A(v_1) \cap A(v_2) & \text{if } A(v_1) \cap A(v_2) \neq \emptyset \\ A(v_1) \cup A(v_2) & \text{otherwise.} \end{cases}$$

Phase (2) proceeds as follows: for a node v whose parent $f(v)$ has already been processed:

$$B(v) = \begin{cases} \sigma \in A(v) \cap A(f(v)) & \text{if } A(v) \cap A(f(v)) \neq \emptyset \\ \sigma \in A(v) & \text{otherwise.} \end{cases}$$

The algorithm above applies only to binary trees. Nonetheless, a straightforward extension to arbitrary k-degree trees can be easily achieved. We now prove a lemma that will be useful later.

Lemma 1. *Let T be a tree and C a single character over the alphabet Σ. Let x be the number of internal nodes v s.t. $|A(v)| > 1$ by applying Fitch's algorithm on (T, C). Then x is less than twice S^*—the parsimony score of T over C.*

Proof. We prove the lemma by induction on l, the length of the path from root r to the closest leaf. Obviously, we are interested only in cases where $|A(r)| > 1$ in the first phase. For $l = 1$, T is a cherry[2] with two leaves v_1 and v_2 with $A(v_1) \cap A(v_2) = \emptyset$ and the lemma follows. Assume correctness for $l = k$ and we prove for $l = k + 1$. We divide the proof into two cases:

- $A(v_1) \cap A(v_2) = \emptyset$: There must be additional mutation from v and the lemma follows.

[2] A *cherry* is a rooted tree with three nodes: the root, and two leaves which are children of the root.

- $|A(v_1)| > 1$ and $|A(v_2)| > 1$: In this case there might be no mutation from v to either of his children (e.g. $A(v_1) = \{A, C, G\}$ and $A(v_2) = \{A, G\}$). Let x_1 and x_2 be the number of nodes w in T_{v_1} and in T_{v_2} resp. with $|A(w)| > 1$, and S_1^* and S_2^* the optimal scores for T_{v_1} and T_{v_2} resp. It is clear that $S^* = S_1^* + S_2^*$, however by the assumption, $x = x_1 + x_2 + 1 < x_1 + 1 + x_2 + 1 \leq 2(S_1^* + S_2^*)$ and the assumption follows.

Problem 1 was extended to phylogenetic networks in [14, 15, 31], and its quality as a criterion for reconstructing and evaluating networks was established on both synthetic and biological data in a series of papers [31, 21, 22].

Definition 1. *Parsimony Score of Phylogenetic Networks (PSPN)*

Input: *A* 3*-tuple* (S, N, λ^k)*, where* N *is a phylogenetic network and* λ^k *is the labeling of* $\mathcal{L}(N)$ *by the sequences in* S*.*
Output: *The extension* $\hat{\lambda}^k$ *that minimizes the expression*

$$\sum_{1 \leq i \leq k} \left[min_{T \in \mathcal{T}(N)} pars(S, T, \lambda^k, i) \right].$$

3 Hardness of Approximation of the PSPN Problem

In [23], we proved that the PSPN problem is NP-hard by a reduction from the max-2-sat problem. By [13], there is a constant ζ such that there is no polynomial time algorithm for max-2-sat with performance ratio better than ζ, *i. e.* there are $P1$ and $P2$ such that $gap - max - 2sat[P1, P2]$[3] is NP-hard (see [16] for the definition of gap problems). Thus by the reduction in [23] there is a constant $\zeta^{'}$ such that there is no polynomial time algorithm for $PSPN$, and $gap - PSPN[4 * |C| - P2 + |U|, 4 * |C| - P1 + |U|]$ is NP-hard.

Corollary 1. *There is a constant* ζ' *such that there is no polynomial time algorithm for PSPN with performance ratio better than* ζ'*.*

Corollary 2. *The PSPN problem is hard to approximate even for networks of bounded degrees, where each node has at most 20 children.*

This result follows from the fact that the $gap - max - 3sat$ problem, when every variable appears 5 times, is hard.

It is important to note that our reduction in [23] generates networks with no more than *one* HGT between any pair of edges. Thus the hardness of approximation results hold also for such networks. In the next section we provide an approximation algorithm for a network with up to one[4] HGT between each pair of edges.

[3] In a $gap - max - 2sat[A, B]$ problem, where $A < B$, a YES-instance is a formula in which at least B clauses are satisfiable, and a NO-instance is a formula in which at most A clauses are satisfiable. If the number of satisfiable clauses is strictly greater than A and strictly smaller than B, then either answer (YES or NO) can be given.

[4] The algorithm can be generalized to the case where the number of HGTs between each pair of edges is bounded by some constant $c > 1$. This will increase the approximation ratio.

4 A Linear-Time Algorithm

Our linear time algorithm builds on the improved heuristic of [21] for the PSPN problem, outlined in Fig. 2. The algorithm is based on the fact that there always exists a *lowest reticulation edge* in a phylogenetic network that satisfies the temporal constraints described in [30]. A reticulation edge $e = (u \rightarrow v)$ is called *a lowest reticulation edge* (or just a lowest edge) if there is no reticulation edge (other than e) adjacent to any node in either T_u or T_v.

```
**ExactPSPN**(N=(V',E'))

1. If $N$ is not a tree
    (a) Find a lowest reticulation edge $e = (u \rightarrow v)$ in $N$;
    (b) Let $e'$ be the edge between $v$ and its ancestral node on the tree edge;
    (c) By Fitch's algorithm, compute the optimal assignment $A$ of $u$ and $v$;
    (d) If $A(u) \cap A(v) = \emptyset$ then
        return $(V', E' \setminus e)$;
    (e) else if $A(u) \subseteq A(v)$ then
        return $(V', E' \setminus e')$;
    (f) else
        i. $opt = pars(ExactPSPN(V', E' \setminus e))$;
        ii. $A(u) \leftarrow A(v)$;                     // update $v'$s values
            $opt' = pars(ExactPSPN(V', E' \setminus e'))$;
        iii. if $opt' < opt$ return $(V', E' \setminus e')$; else return $(V', E' \setminus e)$.
2. else return $Fitch(N)$.
```

Fig. 2. The improved heuristic algorithm

```
**Linear-PSPN**$(N = (V, E))$

1. If $N$ is not a tree
    (a) Find a lowest reticulation edge $e = (u \rightarrow v)$ in $N$;
    (b) Let $e'$ be the edge between $v$ and its ancestral node on the tree edge;
    (c) By Fitch's algorithm compute the optimal bottom up assignment $A$
        to $T_u$ and $T_v$, $A(T_u)$ and $A(T_v)$;
    (d) If $A(u) \cap A(v) = \emptyset$ then
        $\hat{\lambda}_1 =$ Linear-PSPN$(V, E \setminus e)$;
    (e) else
        $\hat{\lambda}_1 =$ Linear-PSPN$(V, E \setminus e')$;
    (f) return $\hat{\lambda}_1$.
2. Continue first phase of Fitch on the tree N without changing internal
   labels that have already determined.
3. Perform second phase of Fitch on the tree N.
4. Return the resultant fully labelled tree.
```

Fig. 3. The Linear-PSPN algorithm

The algorithm in Fig. 2 checks in each step a lowest reticulation edge of the network. It calculates $A(u)$ and $A(v)$ by Fitch's algorithm. In a case where $\neg((A(u) \bigcap A(v) = \emptyset) \vee (A(u) \subseteq A(v)))$ the algorithm considers recursively (and separately) both the reticulation edge and the (alternative) tree edge (i.e. the network with and the network without $(u \rightarrow v)$) . The running time of the algorithm is exponential with the number of such cases.

Our new linear-time algorithm is similar to the exact heuristic algorithm described in Fig. 2 in its recursive style and the search for a lowest reticulation edge at every invocation. However, in contrast, whenever we are unsure of a mutation along that edge, we just take it. Formally, we remove the exponential component from the exact algorithm $PSPN$ and perform step (1e) in any case the condition at step (1d) is not satisfied. The algorithm, Linear-PSPN(N), is outlined in Fig. 3.

Claim. Let $E(N)$ be the set of reticulation and tree edges in N. Then the algorithm terminates and runs in time $O(E(N))$.

4.1 A 3-Approximation Ratio

An algorithm A for a minimization problem P with optimal solution $opt(P)$ (or just *opt* for short), is a polynomial time α-approximation algorithm if A runs in polynomial time and the score of the solution returned by A, $A(P)$, satisfies

$$A(P) \leq \alpha \cdot opt(P).$$

We now show that if the number of reticulation edges emanating from a tree edge is at most one, Linear-PSPN yields a 3-approximation algorithm. The analysis relies on Lemma 1 above.

The technique we use is based on the *local ratio* technique which is useful for approximating optimization covering problems such as vertex cover, dominating set, minimum spanning tree, feedback vertex set and more [4, 2, 3]. The technique recursively solves local sub-problems until a solution is found. The way the local sub-problems are solved determines the approximation ratio. In general, we decompose the network into two networks and show that two *separate* optimal solutions to the networks are a lower bound to an optimal solution to the complete network.

Theorem 1. *If the maximum number of reticulation edges emanating from a tree edge is* 1*, then the approximation ratio of* $Linear - PSPN$ *is 3.*

Proof. We start with a central observation to give a lower bound on the optimal score of a given network.

Observation 1. *Let* $e = (u \rightarrow v)$ *be a lowest reticulation edge in a network* N*. Let* $N' = N \setminus T_v$ *be the network obtained by pruning* T_v *from* N *(including the edges leading to* v*). Then* $opt(N) \geq opt(N') + opt(T_v)$*.*

Proof. Simply take the tree T with the assignment to internal nodes $A(T)$ yielding $opt(N)$ as an upper bound on $opt(N') + opt(T_v)$.

Corollary 3. *If we find an* α *approximation to both* $opt(N')$ *and* $opt(T_v)$*, we find an* α *approximation to* N*.*

We now show how the 3-ratio is obtained. At any local step, we remove a subtree that was solved optimally and contains no reticulation edges (or contains only such edges that did not incur a mutation). This subtree is connected to the rest of the network by a $(u \to v)$ reticulation edge with $A(v) \subset A(u)$. Let T_v be the tree removed from the rest of the network. Such a reticulation edge might incur an additional mutation. However, note that $|A(u)| > 1$. Now, since there is no reticulation edge entering T_u that can reduce the number of mutations, there exists an optimal solution with T_u as a subgraph. By Lemma 1 the number of mutations in T_u is at least half the number of nodes u' with $|A(u')| > 1$. By our assumption, every edge entering such a node u' gives rise to at most one extra mutation. We simply change that extra mutation on u' and the theorem follows. The rest of the network is solved recursively.

5 Experimental Results

We implemented the approximation algorithm and evaluated both its accuracy and execution time through experiments on both simulated and biological datasets. We performed experiments on a 2.4 GHz Intel Pentium 4 PC. Accuracy of the approximation algorithm was measured as the difference of the parsimony scores computed by the approximation algorithm and the exact algorithm normalized by the parsimony score computed by the exact algorithm, presented as percentage. Execution times of both the approximation algorithm and the exact algorithm were measured and speedups of the approximation algorithm over the exact algorithm were reported.

Simulated Datasets. For the simulated datasets, we first used the `r8s` tool [35] to generate a random birth-death phylogenetic tree on 20 taxa. The `r8s` tool generates molecular clock trees; we deviated the tree from this hypothesis by multiplying each edge in the tree by a number randomly drawn from an exponential distribution. The resulting tree was taken as the species tree. The expected evolutionary diameter (longest path between any two leaves in the tree) was 0.2. A model phylogenetic network was generated by adding 5 HGT edges to the model tree.

Based on the model network, we used the Seq-gen tool [34] to evolve 26 datasets of DNA sequences of length 1500 down the "species" tree and DNA sequences of length 500 down the other tree contained inside the network (the one that exhibits all HGT events). Both sequence datasets were evolved under the K2P+γ model of evolution, with shape parameter 1 [25]. Finally, we concatenated the two datasets.

Biological Datasets. We have included experimental results on three biological datasets we previously studied [22]. The first biological dataset is the rubisco gene *rbcL* of a group of 46 plastids, cyanobacteria, and proteobacteria, which was analyzed by Delwiche and Palmer [6]. This dataset consists of 46 aligned amino acid sequences (each of length 532), 40 of which are from Form I of rubisco and the other 6 are from Form II of rubisco. The first 21 and the last 14 sites of the sequence alignment were excluded from the analysis, as recommended by the authors. The species tree for the dataset was created based on information from the ribosomal database project (http://rdp.life.uiuc.edu) and the work of [6]. The second dataset consists of the ribosomal protein *rpl12e* of a group of 14 Archaeal organisms, which was analyzed by Matte-Tailliez *et al.* [28].

This dataset consists of 14 aligned amino acid sequences, each of length 89 sites. The authors constructed the species tree using Maximum Likelihood, once on the concatenation of 57 ribosomal proteins (7,175 sites), and another on the concatenation of SSU and LSU rRNA (3,933 sites). The two trees are identical, except for the resolution of the *Pyrococcus* three-species group; we used the tree based on the ribosomal proteins. The third dataset consists of the ribosomal protein gene *rps11* of a group of 47 flowering plants, which was analyzed by Bergthorsson *et al.* [5]. This data set consists of 47 aligned DNA sequences, each with 456 sites. The authors analyzed the 3' end of the sequences separately; this part of the sequences contains 237 sites. The species tree was reconstructed based on various sources, including the work of [29] and [24].

5.1 Results and Analysis

We evaluated the performance of the algorithms in terms of accuracy and speedup. Since the running time of the exact algorithm for computing the parsimony score of a phylogenetic network is affected by the number of trees that it considers inside the network, we also plotted the average numbers of trees that the exact algorithm considers, so that we understand the gains in speed for the approximation algorithm, which considers exactly one tree in all cases.

Fig. 4 shows the results of the 26 simulated datasets for networks with up to 6 HGT edges. The results were collected from 1000 sampled valid networks for each case of the multiple gene transfers. HGTs in each network are distributed differently. Overall, the approximation algorithm is very accurate with the statistical mean being about 1% different in the parsimony scores computed, compared with the exact algorithm. All parsimony scores computed by the approximation algorithm were within 3.5% of the optimal scores. For the networks with less then 5 HGTs, the approximation algorithm achieves about the same accuracy of the exact algorithm in most of the networks. The figure also shows that the approximation algorithm is up to 70% faster than the exact algorithm, with statistical mean around 32%. The improved execution time of the approximation algorithm came from the fewer number of trees created for computing parsimony score. Fig. 4 also shows the average number of trees that the exact algorithm considers. The average number of trees created increases as the number of HGTs increases. For networks with 6 HGTs (simulated dataset), the average number of trees can be up to 2.

For the rubisco gene *rbcL* dataset, We tested networks with up to 8 HGTs. In each case of the multiple gene transfers, we selected 500 valid networks with HGTs being placed differently. As the results in Fig. 4 show, the approximation algorithm is almost as accurate as the exact algorithm (within 0.5%; see the small boxes or the lower quartile for 7-HGT case at the bottom). Very few outliers exist across different numbers of HGTs. On the other hand, the approximation algorithm performs very efficiently. It performs up to a factor of 7 faster than the exact algorithm. The statistical mean of the improvement increases as the number of HGTs increases, with an exception in the case of 8 HGTs, where the sampled networks are probably not distributed well enough.

Similar trends are observed with the other two biological datasets, as shown in Fig. 4. The figures show that the statistical mean of the difference in accuracy is almost 0 in all cases, which indicates that the approximation algorithm computes almost identical scores as the exact algorithm, in most cases. The speedup factors, and their correlations

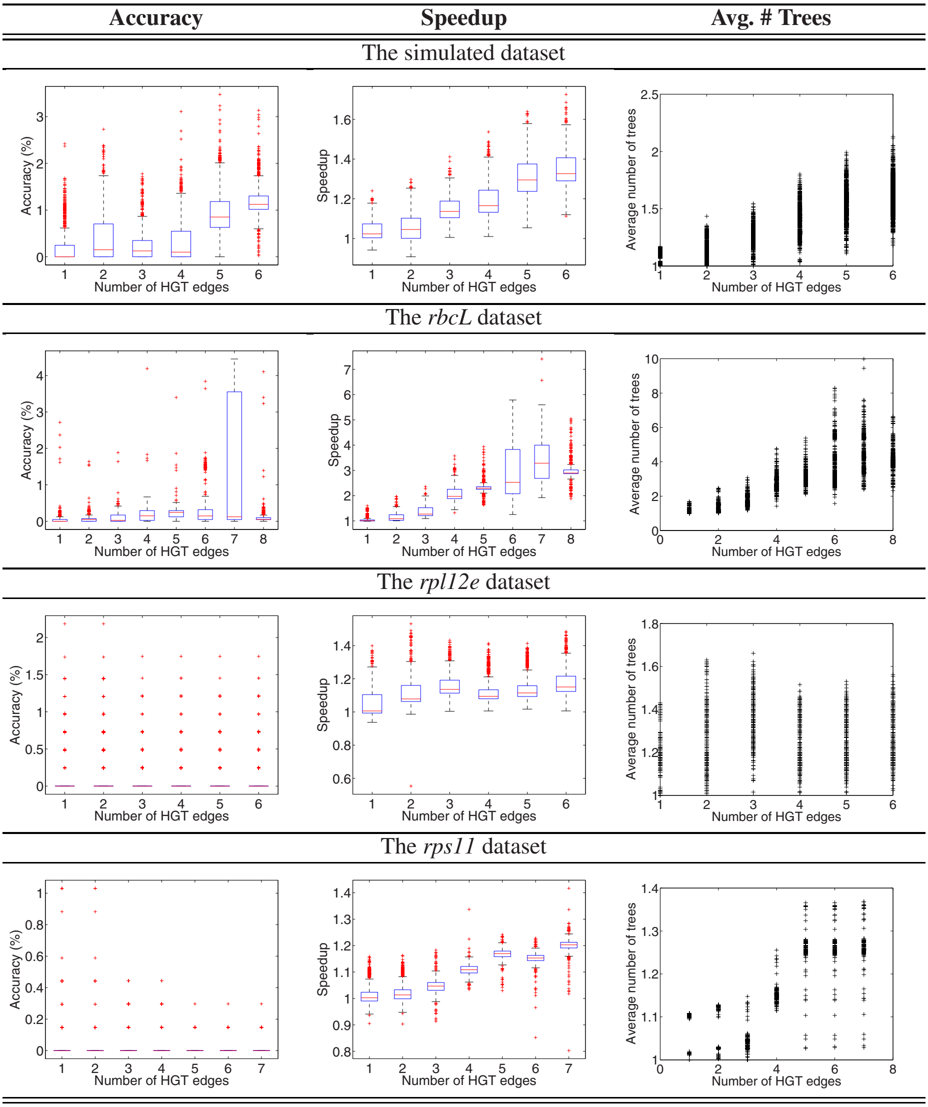


Fig. 4. Results for the four datasets. Accuracy is computed as$((MP_{approx} - MP_{exact})/MP_{exact})$, and shown as percentage. Speedup is computed as the execution time of the exact algorithm divided by the that of the approximation algorithm. The right column shows the average number of trees created for computing parsimony by the exact algorithm.

to the numbers of trees the exact algorithm considers, are also shown, and they show improvements up to a factor of 1.5. We expect that for larger datasets the gains in performance (speedup) will be even more pronounced. If one hopes to detect HGT events in large prokaryotic groups, for example, such a speedup is essential.

Acknowledgments

This work was supported in part by the Rice Terascale Cluster funded by NSF under grant EIA-0216467 and a partnership between Rice University, Intel, and HP. Luay Nakhleh was supported in part by the Department of Energy grant DE-FG02-06ER25734, the National Science Foundation grant CCF-0622037, and the George R. Brown School of Engineering Roy E. Campbell Faculty Development Award. Tamir Tuller was supported by the Edmond J. Safra Bioinformatics program at Tel Aviv University.

References

[1] V. Bafna and V. Bansal. Improved recombination lower bounds for haplotype data. In *Proceedings of the Ninth Annual International Conference on Computational Molecular Biology*, pages 569–584, 2005.

[2] V. Bafna, P. Berman, and T. Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM J. on Discrete Mathematics*, 12:289–297, 1999.

[3] R. Bar-Yehuda. One for the price of two: A unified approach for approximating covering problems. *Algorithmica*, 27:131–144, 2000.

[4] R. Bar-Yehuda and S. Even. A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics*, 25:27–46, 1985.

[5] U. Bergthorsson, K.L. Adams, B. Thomason, and J.D. Palmer. Widespread horizontal transfer of mitochondrial genes in flowering plants. *Nature*, 424:197–201, 2003.

[6] C.F. Delwiche and J.D. Palmer. Rampant horizontal transfer and duplication of rubisco genes in eubacteria and plastids. *Mol. Biol. Evol*, 13(6), 1996.

[7] W.F. Doolittle, Y. Boucher, C.L. Nesbo, C.J. Douady, J.O. Andersson, and A.J. Roger. How big is the iceberg of which organellar genes in nuclear genomes are but the tip? *Phil. Trans. R. Soc. Lond. B. Biol. Sci.*, 358:39–57, 2003.

[8] J.A. Eisen. Assessing evolutionary relationships among microbes from whole-genome analysis. *Curr. Opin. Microbiol.*, 3:475–480, 2000.

[9] I.T. Paulsen *et al.* Role of mobile DNA in the evolution of Vacomycin-resistant Enterococcus faecalis. *Science*, 299(5615):2071–2074, 2003.

[10] W. Fitch. Toward defining the course of evolution: minimum change for a specified tree topology. *Syst. Zool*, 20:406–416, 1971.

[11] D. Gusfield and V. Bansal. A fundamental decomposition theory for phylogenetic networks and incompatible characters. In *Proceedings of the Ninth Annual International Conference on Computational Molecular Biology*, pages 217–232, 2005.

[12] M. Hallett, J. Lagergren, and A. Tofigh. Simultaneous identification of duplications and lateral transfers. In *Proceedings of the Eighth Annual International Conference on Computational Molecular Biology*, pages 347–356, 2004.

[13] J. Hastad. Some optimal inapproximability results. *STOC97*, pages 1–10, 1997.

[14] J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.

[15] J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.

[16] D.S. Hochbaum. *Approximation Algorithms for NP-Hard Problems*. PWS Publishing Company, 1997.

[17] D.H. Huson, T. Klopper, P.J. Lockhart, and M. Steel. Reconstruction of reticulate networks from gene trees. In *Proceedings of the Ninth Annual International Conference on Computational Molecular Biology*, pages 233–249, 2005.

[18] T.N.D. Huynh, J. Jansson, N.B. Nguyen, and W.K. Sung. Constructing a smallest refining galled phylogenetic network. In *Proceedings of the Ninth Annual International Conference on Computational Molecular Biology*, pages 265–280, 2005.

[19] R. Jain, M.C. Rivera, J.E. Moore, and J.A. Lake. Horizontal gene transfer in microbial genome evolution. *Theoretical Population Biology*, 61(4):489–495, 2002.

[20] R. Jain, M.C. Rivera, J.E. Moore, and J.A. Lake. Horizontal gene transfer accelerates genome innovation and evolution. *Molecular Biology and Evolution*, 20(10):1598–1602, 2003.

[21] G. Jin, L. Nakhleh, S. Snir, and T. Tuller. Efficient parsimony-based methods for phylogenetic network reconstruction. *Bioinformatics*, 23:e123–e128, 2006.

[22] G. Jin, L. Nakhleh, S. Snir, and T. Tuller. Inferring phylogenetic networks by the maximum parsimony criterion: A case study. *Molecular Biology and Evolution*, 24(1):324–337, 2007.

[23] G. Jin, L. Nakhleh, S. Snir, and T. Tuller. On approximating the parsimony score of phylogenetic networks. Under review, 2007.

[24] W.S. Judd and R.G. Olmstead. A survey of tricolpate (eudicot) phylogenetic relationships. *American Journal of Botany*, 91:1627–1644, 2004.

[25] M. Kimura. A simple method for estimating evolutionary rates of base substitutions through comparative studies of nucleotide sequences. *Journal of Molecular Evolution*, 16:111–120, 1980.

[26] C.R. Linder, B.M.E. Moret, L. Nakhleh, and T. Warnow. Network (reticulate) evolution: biology, models, and algorithms. In *The Ninth Pacific Symposium on Biocomputing (PSB)*, 2004. A tutorial.

[27] V. Makarenkov, D. Kevorkov, and P. Legendre. Phylogenetic network reconstruction approaches. *Applied Mycology and Biotechnology (Genes, Genomics and Bioinformatics)*, 6, 2005. To appear.

[28] O. Matte-Tailliez, C. Brochier, P. Forterre, and H. Philippe. Archaeal phylogeny based on ribosomal proteins. *Molecular Biology and Evolution*, 19(5):631–639, 2002.

[29] F.A. Michelangeli, J.I. Davis, and D.Wm. Stevenson. Phylogenetic relationships among Poaceae and related families as inferred from morphology, inversions in the plastid genome, and sequence data from mitochondrial and plastid genomes. *American Journal of Botany*, 90:93–106, 2003.

[30] B.M.E. Moret, L. Nakhleh, T. Warnow, C.R. Linder, A. Tholse, A. Padolina, J. Sun, and R. Timme. Phylogenetic networks: modeling, reconstructibility, and accuracy. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):13–23, 2004.

[31] L. Nakhleh, G. Jin, F. Zhao, and J. Mellor-Crummey. Reconstructing phylogenetic networks using maximum parsimony. *Proceedings of the 2005 IEEE Computational Systems Bioinformatics Conference (CSB2005)*, pages 93–102, August 2005.

[32] L. Nakhleh, T. Warnow, and C.R. Linder. Reconstructing reticulate evolution in species: theory and practice. In *Proceedings of the Eighth Annual International Conference on Computational Molecular Biology*, pages 337–346, 2004.

[33] C.T. Nguyen, N.B. Nguyen, W.K. Sung, and L. Zhang. Reconstructing recombination network from sequence data: The small parsimony problem. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2006.

[34] A. Rambaut and N.C. Grassly. Seq-gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comp. Appl. Biosci.*, 13:235–238, 1997.

[35] M. Sanderson. `r8s` software package. Available from http://loco.ucdavis.edu/r8s/r8s.html.

[36] D. Sankoff. Minimal mutation trees of sequences. *SIAM Journal on Applied Mathematics*, 28:35–42, 1975.

A Bootstrap Correspondence Analysis for Factorial Microarray Experiments with Replications

Qihua Tan[1,2,*], Jesper Dahlgaard[3], Basem M. Abdallah[4], Werner Vach[5], Moustapha Kassem[4], and Torben A. Kruse[1]

[1] Department of Biochemistry, Pharmacology and Genetics, Odense University Hospital, Sdr. Boulevard 29, DK-5000, Odense C, Denmark
qihua.tan@ouh.fyns-amt.dk
[2] Epidemiology Unit, Institute of Public Health, University of Southern Denmark
[3] Department of Hematology, Aarhus University Hospital, Aalborg, Denmark
[4] Department of Endocrinology, Odense University Hospital, Denmark
[5] Department of Statistics, University of Southern Denmark, Denmark

Abstract. Characterized by simultaneous measurement of the effects of experimental factors and their interactions, the economic and efficient factorial design is well accepted in microarray studies. To date, the only statistical method for analyzing microarray data obtained using factorial design has been the analysis of variance (ANOVA) model which is a gene by gene approach and relies on multiple assumptions. We introduce a multivariate approach, the bootstrap correspondence analysis (BCA), to identify and validate genes that are significantly regulated in factorial microarray experiments and show the advantages over the traditional method. Applications of our method to two microarray experiments using factorial have detected genes that are up or down-regulated due to the main experimental factors or as a result of interactions. Model comparison showed that although both BCA and ANOVA capture the main regulatory profiles in the data, our multivariate approach is more efficient in identifying genes with biological and functional significances.

1 Introduction

As a high-throughput technique, microarray capable of simultaneously measuring mRNA levels for thousands of genes is becoming an increasingly important tool for researchers in biomedical science. At the same time, interpreting the large amount of data produced in microarray experiments imposes a major challenge to bioinformaticians (Lander, 1999). Among the major issues in data analysis is identifying genes that are regulated in a biological process (for example cell cycle, treatment response, disease development) in high dimensional microarray experiments. This is especially challenging for microarray studies using complex experiment designs due to the intricate relationships both between and within the multiple genetic and experimental factors including interactions which can not be predefined.

[*] Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 73–84, 2007.

Factorial experiment design (FED), characterized by simultaneous measurement of the effects of multiple experimental factors (the main effects) and their interactions, is an economic yet efficient complex design popular in use in biomedical studies (Shaw et al., 2002). The nice features of FED have also made it well accepted in microarray experiments (Wildsmith et al., 2001; Yang and Speed, 2002; Churchill, 2002; Glonek and Solomon, 2004). At the same time, statistical methods that take into account the experimental complexity are demanding for dealing with data produced in factorial microarray experiments. Kerr et al. (2000) and Pavlidis (2003) applied the analysis of variance model (ANOVA) to factorial microarray data using the parametric linear regression approach assuming (1) normality in the log intensity of gene expressions and (2) linear relationship between log intensity and the effects of main experimental factors as well as their interactions. In their approaches, each gene is analyzed separately and statistical procedures applied to correct for multiple testing. As usual, multiple replicates are required to ensure model identifiability.

The singular value decomposition (SVD) (Alter et al., 2000) and SVD-based multivariate statistical methods, for example, principal components analysis (Holter et al., 2000; Wall et al. 2003) and correspondence analysis (CA) (Fellenberg et al., 2001; Tan et al., 2004; Baty et al. 2006) have been applied in analyzing multidimensional microarray data. Although such exploratory methods can be used for dimension reduction and for pattern discovery through data visualization, validity of the clusters or identified genes has rarely been examined. In another development, computer intensive resampling methods such as bootstrap (Efron, 1979) are expanding their uses in microarray studies (Kerr et al, 2000; Kerr and Churchill, 2001; Ghosh, 2002; Tan et al. 2004) due to their non-parametric nature. Recently, the bootstrap resampling method has been applied in constructing confidence intervals and in making statistical inferences (Efron, 1981; Bhamre et al. 2000; Wood, 2005). By bootstrapping the gene contributions on the reduced dimensions, we introduce a novel bootstrap correspondence analysis (BCA) by combining the bootstrap resampling method with CA to identify and validate differentially expressed genes in factorial microarray experiments. Application of our new approach is illustrated by analyzing two factorial microarray datasets from stem cell and breast cancer studies. Results from our analysis will be compared with that from ANOVA to show the advantages of our approach over the traditional method.

2 Methods

2.1 Correspondence Analysis

In microarray experiments using a factorial design, we are actually facing a sophisticated situation where we are interested not only in the effects of the multiple factors but also in the effects of their interactions. The idea of applying CA to FED data is that main effects of the multiple factors together with their interactions which dominate the variance in the data can be captured by the reduced dimensions in the transformed data space.

Suppose in a factorial microarray experiment, there are two experimental factors A and B with p levels in A and q levels in B. Then there will be pxq hybridizations each representing an interactive variable (Clausen, 1988) or combination of experimental factors in the design. If, after gene filtering, we have a total of *n* genes, the data can be summarized by a large *n*x(*p*x*q*) matrix with *n* stands for the number of rows (genes) and *p*x*q* for the number of columns (hybridizations or interactive variables). To carry out CA, we divide each entry in the matrix by the total of the matrix so that the sum of all the entries in the resulted matrix equals 1. We denote the new matrix by P and its elements by p_{ijk} (i stands for the genes from 1 to *n*, j for the levels of factor *A* from 1 to *p* and likewise, *k* for the levels of factor *B* from 1 to *q*). In matrix *P*, the sum of row *i*, $p_{i.} = \sum_j \sum_k p_{ijk}$, is the mass of row *i* and the sum of the column representing the interactive variable A_jB_k, $p_{.jk} = \sum_i n_{ijk}$, is the mass of that column. With the row and column masses, we derive a new matrix *C* with elements $c_{ijk} = (p_{ijk} - p'_{ijk}) / \sqrt{p'_{ijk}}$ where $p'_{ijk} = p_{i.}p_{.jk}$ is the expected value for each element in matrix *P*. By submitting matrix *C* to SVD, we get $C = U\Lambda V'$ where *U* is the eigenvectors of CC', *V* is the eigenvectors of $C'C$, Λ is a diagonal matrix containing the ranked eigenvalues of *C*, λ_l (l=1, 2, ….*p*x*q*). Since the total inertia $\sum_l \lambda_l^2$ equals the sum of c_{ijk}^2 in *C*, the major variance in the original data is captured by the dimensions corresponding to the top elements in Λ.

One big advantage of CA is that, we can simultaneous project genes and interactive variables into a new space with the projection of gene *i* on axis *l* calculated as $g_{il} = \lambda_l u_{il} / \sqrt{p_{i.}}$ where u_{il} is the *i*-th row and the *l*-th column in *U*, and similarly the projection of A_jB_k along axis *l* is $h_{jkl} = \lambda_l v_{jkl} / \sqrt{p_{.jk}}$ where v_{jkl} is the element in the *l*-th column in *V* that corresponds to A_jB_k. In practice, a biplot (Gabriel and Odoroff, 1990) is used to display the projections. The biplot is very useful for visualizing and inspecting the relationships between and within the genes and the interactive variables. In the biplot, genes projected to a cluster of interactive variables associated with one experimental factor are up-regulated due to that factor. Especially, genes projected to a single or standing-alone interactive variable are highly expressed as a result of interaction between the corresponding main factors. As the inertia along the *l*-th axis can be decomposed into components for each gene, i.e. $\lambda_l^2 = \sum_i p_{i.} g_{il}^2$, we can calculate the proportion of the inertia of the *l*-th axis explained by the *i*-th gene as, $ac_{il} = p_{i.} g_{il}^2 / \lambda_l^2$ which is the absolute contribution of the *i*-th gene to the *l*-th axis. The sum of ac_{il} for a group of selected genes stands for the proportion of the total variance explained by these particular genes. If all the *n* genes are randomly distributed along the axis, the null contribution (random mean) by each gene would be expected as $1/n$. The random mean contribution will be used for significance inferences for the observed gene contribution in the next section.

2.2 Non-parametric Bootstrapping

Although genes with large contribution are highly regulated by the experimental factors and their interactions represented by the top dimensions in CA, directly picking up genes with large contribution ignores variability in each of the estimated contributions and is thus unreliable. Ghosh (2002) introduced a resampling method to SVD analysis of time-course data to bootstrap the variability of the modes that characterize the time-course patterns in microarray data. Here we combine non-parametric bootstrap with CA to assess the distribution of the estimated gene contributions to the leading dimensions that feature the effects of main factors as well as the effects arising from their interactions and make statistical inference based on these distributions. When there are *w* replicates available, we randomly pick up with replacement *w* arrays for each interactive variable to form a bootstrap sample of gene expression values which is of the same size as the original sample. The bootstrap distributions of the contributions on a dimension by each gene are obtained by repeating the bootstrapping for *B* times. Based on the distributions, we obtain the bootstrap p-value for comparing the estimated contributions with the random mean as $p \equiv \sum_{t=1}^{B} I(ac_t \leq ac_o)/B$ where $I(\cdot)$ is an indicator function, ac_t is the absolute contribution estimated for each gene in bootstrap sample t and ac_o is the mean random contribution. Here the p-value is obtained by comparing the bootstrap distribution of the estimated contribution by each gene to the random mean with a null hypothesis that the mean of the estimated gene contribution is not higher than the random mean (ac_o). Note that since we are restrictively resampling the replicate arrays for each interactive variable, the functional dependency among the genes are preserved in the bootstrap samples.

2.3 The Analysis of Variance Model

We also analyze the same data using the existing parametric approach, i.e. ANOVA model, with aim at comparing the performances of the two methods. In the analysis, we fit the expression level of a gene (E) as a linear function of the cell line effect (C), vitamin D treatment effect (D) and their interaction (C·D) (Pavlidis, 2003), i.e. we fit

$$E = \mu + C + D + C \cdot D + \varepsilon$$

where μ is the mean expression level of the gene, ε is the random error. In this model, in addition to the linear assumption, the main as well as the interaction effects are assumed to be independent; the random error or the residual ε is normally distributed which is to say that E has a normal distribution. Note that for each of the genes, the model independently tests the main effects and their interaction.

2.4 Functional Analysis of Significant Genes

We apply the EASE software (http://david.niaid.nih.gov/david/ease.htm) to functionally classify and to conduct category over-representation analysis (COA) of the differentially expressed genes. EASE calculates the one-tailed Fisher's exact probability for over-representation of a functional category by using the Gaussian hypergeometric probability distribution of randomly sampling a given number of genes and observing a specific number belonging to the classification (Hosack et al. 2003). COA is applied to significant genes identified from BCA and ANOVA with aim at comparing their performances in producing meaningful results with biological significances.

3 Applications

3.1 Stem Cell Data

We use data from a microarray experiment (using Affymetrix HG-U133A 2.0 chips each containing 22,000 genes) on stem cells conducted in our lab. In the experiment, two lines of human mesenchymal stem cells (hMSC), telomerase-immortalized hMSC (hMSC-TERT) and hMSC-TERT stably transduced with the full length human delta-like 1 (Dlk1)/Pref-cDNA (hMSC-dlk1), were treated with vitamin D to examine the effects of Dlk1, vitamin D and their interaction on hMSC growth and differentiation and to look for genes that are differentially expressed in the experiment. The experiment was done using a 2x2 factorial design. Twelve hybridizations in total were conducted with each of the four interactive variables in triplicates: hMSC-TERT untreated by vitamin D or tert-control (designated as tC), hMSC-TERT treated with vitamin D (tD), hMSC-dlk1 untreated with vitamin D or dlk-control (dC), hMSC-dlk1 treated with vitamin D (dD). The data can be requested by contacting: babdallah@ health.sdu.dk. The raw data (at probe level) were normalized using the invariant-set normalization method and the intensities for the probes in each probe-set summarized using the model-based gene expression indexes (Li and Wong, 2001a,b) by applying the free dChip software for Affymetrix arrays (http://www.biostat.harvard.edu/ complab/dchip). Finally, genes are filtered by dropping those whose expressions failed to vary across the hybridizations or arrays (standard deviation/mean<0.2) and whose expression index is less than 20 in any of the arrays which resulted in 3,381 genes for subsequent analyses.

The biplots from the correspondence analysis of our stem cell data is shown in Figure 1 where projections of both the genes and the four combinatory or interactive variables (between cell lines and vitamin D treatments, the suffix number indicates replicate) along the first dimension or axis are plotted against that along the second (Figure 1A) and along the third dimension against the fourth (Figure 1B). In Figure 1A the first axis, which accounts for 57.92% of the total variance, separates the two cell lines. It is interesting to see that both tC and tD are projected to the left panel and closely coordinated near the first axis while both dC and dD are projected to the right but with a large distance spanning them. It is easy to find that the second axis (accounting for 24.24% of the total variance) mainly represents the effect of vitamin D treatment in the hMSC-dlk1 cell line. The overall information carried by Figure 1A is that (1) there are genes that are differentially regulated in the two lines (main effect); (2) vitamin D treatment has only trivial effect in the hMSC-TERT cell line; (3) vitamin

D treatment effect is mainly observed in the hMSC-dlk1 cell line (a cell line specific effect or interaction). Unlike Figure 1A, inspection on Figure 1B does not reveal any biological significance. This is understandable because the third and fourth axes explain only 5.99% and 4.94% of the total variance. Since the variance in the data is overwhelmingly dominated by the first and the second axes, Figure 1 reveals that significance in the experiment is represented firstly by genes differentially expressed in the two cell lines, and secondly by genes regulated in response to vitamin D treatment in the hMSC-dlk1 cell line. In addition, note that our gene filtering procedure has left a hole in the cloud of genes in the center of Figure 1A.

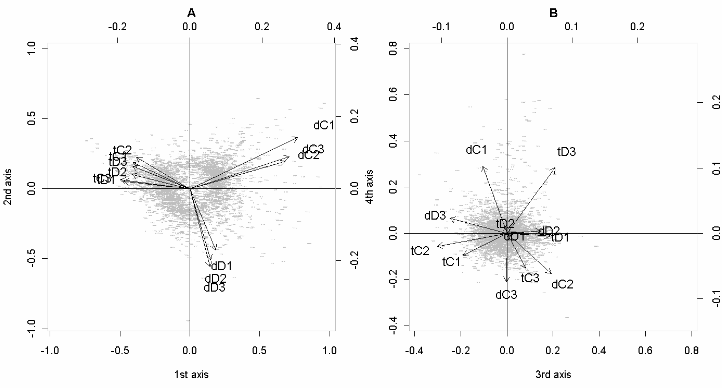


Fig. 1. Biplots showing the projections by both genes and the interactive variables (samples) on the first against that on the second axes (1A) and the third against the fourth axes (1B)

We use the described bootstrap procedure to obtain the empirical distributions of gene contribution on the first two axes and to make significance inferences by comparing with the mean random contribution (1/3,381=0.0003). By setting B to 100,000, we find highly significant genes (p<0.00001) that contribute to the first (352 genes) and the second (221 genes) axes. These genes explain 56.07% and 42.5% of the total variance along each of the two axes. Figure 2 is a heatmap displaying the expression profiles for genes highly significantly contributing to the first (2A) and the second (2B) axes. Obviously Figure 2A has genes showing differential expressions in the two cell lines. Consistent with Figure 1, the upper gene cluster in Figure 2A (indicated by U) are strikingly highly expressed in the hMSC-dlk controls. Figure 2B is mainly characterized by genes up-regulated in the vitamin D treated hMSC-dlk cells (upper gene cluster, U), in the hMSC-dlk controls (middle gene cluster, M) and genes down-regulated in the vitamin D treated hMSC-dlk cells (lower gene cluster, L), all of which representing interaction effects. From Figure 2B, we can also see that transcriptional activities in the hMSC-TERT cell line are unaffected by vitamin D treatment (right panel of Figure 2B) as indicated by Figure 1A.

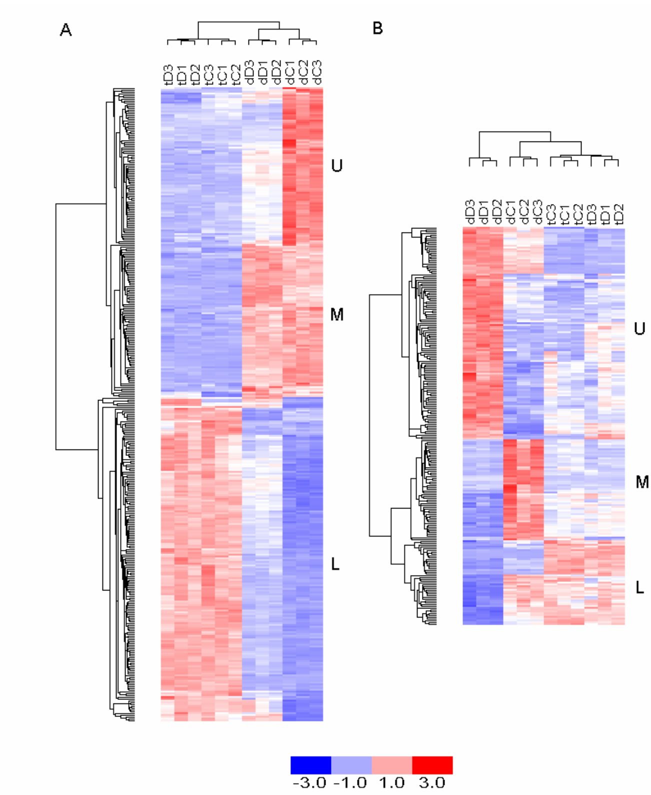


Fig. 2. Heatmaps showing the differential gene expression profiles for the BCA selected significant genes regulated by the cell line effect (2A) and by vitamin D treatment (2B)

We continue our analysis on the same data but using the ANOVA model as described in the method section. The analysis detected highly significant genes ($p<0.00001$) that are differentially expressed between the two cell lines (572 genes), between vitamin D treated and untreated groups (127 genes) and as a result of interaction (77 genes). The selected 572 significant genes showing cell line effect contribute to 53.13% and 25.26% of the total variances along the first two axes in BCA. The 127 genes regulated by vitamin D treatment contribute to 10.13% and 17.93%, and the 77 genes exhibiting interaction effect contribute to 9.16% and 10.3% of the total variance over the first two axes respectively. Overall, differentially expressed genes in the two cell lines identified by the ANOVA model mainly contribute to the first axis and significant vitamin D regulated genes to the second axis. The 77 interaction regulated genes contribute to the first and the second axes nearly equally but with only a small amount. Note that, there are 62 out of the 127 vitamin D affected genes (49%) and 40 out of the 77 interaction affected genes (52%) overlap with the genes showing cell line effect in ANOVA. Also among the 77 interaction regulated genes, 51 overlap with the 127 vitamin D affected genes (66%).

In contrast, among the 352 and the 221 genes that significantly contribute to the first two axes in BCA, only 49 overlapped.

Finally we conduct a functional analysis of the top significant genes differentially regulated in the two cell lines using EASE. To do that, we select 100 significant genes with highest contribution to the first axis in BCA and 100 genes with highest significance (p<0.0000001) on the cell line effect in ANOVA. There are 39 genes overlap in the two gene lists. For these genes, EASE identified 69 functional categories for the genes from BCA and 84 categories for genes from ANOVA indicating a higher functional diversity in the ANOVA gene list. COA found a total of 13 functional categories with p<0.05 in the BCA gene list while only one significant category (receptor binding, p=0.007) in the ANOVA gene list. In Table 1, we show the Bonferroni corrected p-values for the 13 functional categories. The results indicate that significant genes detected by BCA are biologically more informative and meaningful than that found by ANOVA.

Table 1. Functional analysis of top 100 significant genes showing cell line effect discovered by BCA and ANOVA

Functional category of genes	Bonferroni corrected p-value	
	BCA	ANOVA
Response to external stimulus	8.63e-006	9.11e-001
Cellular process	2.40e-004	8.28e-001
Receptor binding	9.35e-004	6.46e-003
Cell communication	3.10e-003	1
Response to chemical substance	4.01e-003	1
Inflammatory response	8.85e-003	1
Innate immune response	1.13e-002	1
Response to pest/pathogen/parasite	1.15e-002	1
Response to abiotic stimulus	1.15e-002	1
Response to biotic stimulus	1.60e-002	1
Response to wounding	1.63e-002	1
Cytokine activity	1.85e-002	5.85e-002
Signal transduction	4.18e-002	1

3.2 Breast Cancer Data

As an independent example, we further apply our approach to a breast cancer dataset using the factorial design. The data was analyzed by Scholtens et al. (2004) using linear modeling. The data we use here is provided by *factDesign* which is a R package for analyzing factorial experiment data in Bioconductor (http://www.bioconductor.org). The experiment was aimed at assessing the effect of estrogen on gene expression in ER+ breast cancer cells over time. The four combinations of the two experimental factors (estrogen: exposed and unexposed; time: 10 and 48 hours) represents a typical 2x2 factorial experiment design. The data available from *factDesign* contains expression measurements for 500 genes in 8 samples (2 replicates in each of the combinations). Here we apply BCA to the same data to look for genes affected by the main and the interaction effects. In Figure 3, we show that the data is characterized by the main effects of time (T for 48 and t for 10 hours) presented by the first axis and estrogen (E for exposed and e for unexposed) by

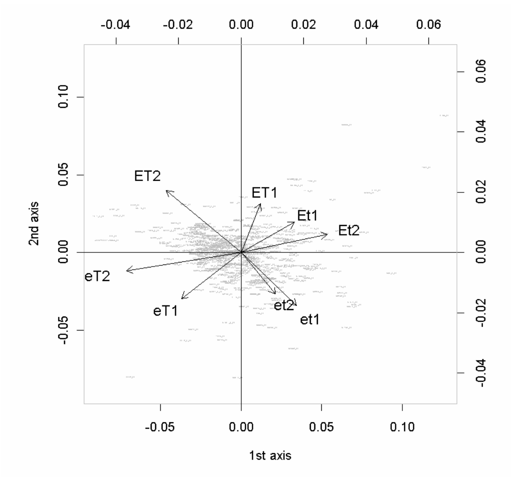


Fig. 3. Biplot showing the CA analysis of estrogen data. The data is characterized by the main effects of time (T for 48 and t for 10 hours) presented by the first axis and estrogen (E for exposed and e for unexposed) by the second axis (the number indicates replicates).

Table 2. Comparison of EASE scores for the two 27 gene sets identified by BCA and ANOVA

Functional category of genes	EASE score		Bonferroni corrected p-value	
	BCA	ANOVA	BCA	ANOVA
Nucleus	1.00e-003	5.01e-001	2.51e-001	1
Nucleobase\, nucleoside\, nucleotide and nucleic acid metabolism	2.77e-003	4.58e-001	6.92e-001	1
Obsolete cellular component	8.07e-003	4.05e-001	1	1
Transcription from Pol II promoter	8.69e-003	4.13e-001	1	1
Chromosome	9.45e-003	2.31e-001	1	1

the second axis (the number indicates replicates) with the first two axes accounting for 78.61% of the total variance in the data. It is interesting to see that, Figure 3 also shows that there are genes specifically regulated in the estrogen unexposed samples at 10 hours (down-right panel) and at 48 hours (down-left panel) which are interaction effects between time and estrogen exposure. Data analysis using *factDesign* in its manual identified 27 genes that are differentially regulated by estrogen after both 10 and 48 hours. By applying BCA (B=1,000), we identified 31 genes as showing estrogen effect (p<0.05). For comparison purpose, we pick up the top 27 genes with estrogen effect from BCA (equivalent to p<0.02). Of the 27 selected genes by each method, nearly half overlap (13 genes). The 27 genes selected by BCA and by *factDesign* are submitted to EASE for functional analysis. The results are shown in Table 2 where no functional category is significant after Bonferroni correction for genes selected by *factDesign* while 2 categories from BCA genes are still significant or nearly significant. Table 2 also presents the EASE score which can be regarded as an uncorrected p-value. Again, the results indicate that BCA genes are more significantly presented into functional categories than genes selected by *factDesign*.

4 Discussion

We have presented a bootstrap correspondence analysis for analyzing high-dimensional microarray data produced in factorial experiments. Application to stem cell and breast cancer data using FED has shown that the method is capable of capturing both the main effects as well as the interaction effects. Our comparative study shows that BCA produces more meaningful results (or gene list) that can be interpreted to significant biological schemes (functional categories) than the traditional ANOVA model. Our approach is characterized by the following features:

1. As a non-parametric method, it is free from the assumptions in the parametric ANOVA model (i.e. normality of gene expression and linear relationship between the level of gene expression and the covariates). Furthermore, our non-parametric bootstrap procedure does not rely on any parametric distribution in making significance inferences.
2. As a multivariate approach, correspondence analysis simultaneously casts and coordinates the genes and the samples onto new dimensions and calculates their contributions. This operating characteristic is again in contrast with the traditional ANOVA model which analyzes each gene separately and ignores the integrated and complex nature in functional genomics. Moreover, in our bootstrap resampling procedure, the inherent functional dependency among the genes is kept intact. As a result, more biologically interpretable results are produced by BCA as compared with ANOVA (Tables 1 and 2).
3. As an unsupervised approach, our method identifies genes significantly contributing to the top dimensions which dominate the variances in the data. As the observed variances can be ascribed to experimental factors as well as their interactions, important genes can be captured by BCA based on their contributions to the major dimensions. This is further in contrast with and advantageous over the supervised parametric ANOVA model because, in reality, a black-and-white assertion may not always hold in describing a biological phenomenon. The very high proportion of overlapping significant genes between the factors (including interaction) is a good example. Such a situation also inflates the multiple testing problem which is a big issue in microarray data analysis.
4. As a data visualization method, correspondence analysis graphically displays the intricate relationship between the genes and the experimental factors (Fellenberg et al. 2001) through the use of a biplot. As such, the combination of bootstrap method with CA allows investigators to integrate data visualization with significance inferences and to focus on important interactive variables with more efficiency. For example, from Figure 1, one can clearly see that vitamin D treatment only affects gene expression

in the hMSC-dlk1 cell line and has no effect in the hMSC-TERT cell line. The cell line effect on gene expression is mainly dominated by genes differentially expressed between the hMSC-TERT cells and the vitamin D treated hMSC-dlk1 cells. Meanwhile, the vitamin D treatment effect is purely characterized by the differential gene expression in the hMSC-dlk1 cell line conditional on vitamin D treatment. In this case, forcing the model to estimate a pure cell line or treatment effect is inappropriate. Such biased modeling can be responsible for the high overlapping proportion in the lists of significant genes identified by ANOVA and for their low representation in functional analysis (Table 1).

5. As a dimension reduction approach, extending BCA to high order FED microarray data is just straight forwards without sacrificing too much for the increased number of parameters as in the ANOVA model. Also, the same procedure can be applied to other types of microarray experiment design, for example, the case-control or time-course studies, to look for important genes that are of biological significances.

Acknowledgements

This work was jointly supported by the Danish Biotechnology Instrument Center (DABIC) under the biotechnological research program of the Danish Research Agency; the Clinical Institute at OUH; and the Danish Center for Stem Cell Research.

References

Alter O., Brown P.O., Botstein D. (2000) Singular value decomposition for genome-wide expression data processing and modeling. *Proc Natl Acad Sci U S A.*, **97**, 10101-10106.

Bhamre S., Nuzzo R.L., Whitin J.C., Olshen R.A. Cohen H.J. (2000) Intracellular reduction of selenite into glutathione peroxidase. Evidence for involvement of NADPH and not glutathione as the reductant. *Molecular and Cellular Biochemistry*, **211**, 9-17.

Baty F., Facompre M., Wiegand J., Schwager J., Brutsche M.H. (2006) Analysis with respect to instrumental variables for the exploration of microarray data structure. *BMC Bioinformatics*, **7,** 422.

Churchill G.A. (2002) Fundamentals of experimental design for cDNA microarrays. *Nat Genet.*, **32**, S490-S495.

Clausen S.E. (1988) Applied correspondence analysis: An introduction. Sage publications.

Efron B. (1979) Bootstrap methods: Another look at the jackknife. *Ann. Statis*t. **7**, 1-26.

Efron B. (1981) Nonparametric estimates of standard error: The Jackknife, the Bootstrap and Other Methods. *Biometrika*, **68**, 589-599.

Fellenberg K., Hauser N.C., Brors B., Neutzner A., Hoheisel J.D., Vingron M. (2001) Correspondence analysis applied to microarray data. *Proc Natl Acad Sci U S A.*, **98,** 10781-10786.

Gabriel K.R., Odoroff C.L. (1990) Biplots in biomedical research. *Stat Med.*, **9,** 469-485.

Ghosh D. (2002) Resampling methods for variance estimation of singular value decomposition analyses from microarray experiments. *Funct Integr Genomics.*, **2,** 92-97.

Glonek G.F., Solomon P.J. (2004) Factorial and time course designs for cDNA microarray experiments. *Biostatistics*, **5**, 89-111.

Holter N.S., Mitra M., Maritan A., Cieplak M., Banavar J.R., Fedoroff N.V. (2000) Fundamental patterns underlying gene expression profiles: simplicity from complexity. *Proc Natl Acad Sci U S A.*, **97,** 8409-8414.

Hosack D.A., Dennis Jr G., Sherman B.T., Lane H.C., Lempicki R.A. (2003) Identifying biological themes within lists of genes with EASE. *Genome Biology*, **4,** R70.

Kerr M.K., Martin M., Churchill G.A. (2000) Analysis of variance for gene expression microarray data. *J Comput Biol.*, **7,** 819-837.

Kerr M.K., Churchill G.A. (2001) Bootstrapping cluster analysis: assessing the reliability of conclusions from microarray experiments. *Proc Natl Acad Sci U S A.*, **98,** 8961-8965.

Lander E.S. (1999) Array of hope. *Nat Genet.* **21,** S3-S4.

Li C., Wong W.H. (2001a) Model-based analysis of oligonucleotide arrays: model validation, design issues and standard error application. *Genome Biology*, **2**, research0032.1-0032.11

Li C., Wong W.H. (2001b) Model-based analysis of oligonucleotide arrays: expression index computation and outlier detection. *Proc Natl Acad Sci U S A.* **98**, 31-36.

Pavlidis P. (2003) Using ANOVA for gene selection from microarray studies of the nervous system. *Methods.*, **31,** 282-289.

Scholtens D., Miron A., Merchant F.M., Miller A., Miron P.L., Iglehart D., Gentleman R. (2004) Analyzing factorial designed microarray experiments. *Journal of Multivariate Analysis*, **90**, 19-43.

Shaw R., Festing M.F., Peers I., Furlong L. (2002) Use of factorial designs to optimize animal experiments and reduce animal use. *ILAR J.*, **43,** 223-232.

Tan Q., Brusgaard K., Kruse T.A., Oakeley E., Hemmings B., Beck-Nielsen H., Hansen L., Gaster M. (2004) Correspondence analysis of microarray time-course data in case–control design, *Journal of Biomedical Informatics,* **37,** 358-365.

Wall M.E., Rechtsteiner A., Rocha L.M. (2003) Singular value decomposition and principle component analysis. *In* A Practical Approach to Microarray Data Analysis (D.P. Berrar, W. Dubitzky, M. Granzow, eds.) *Kluwer: Norwell, MA.* pp.91-109.

Wildsmith S.E., Archer G.E., Winkley A.J., Lane P.W., Bugelski P.J. (2001) Maximization of signal derived from cDNA microarrays. *Biotechniques*, **30**, 202-208.

Wood M. (2005) Bootstrapped confidence intervals as an approach to statistical inference. *Organizational Research Methods*, **8**, 454-470.

Yang Y.H., Speed T. (2002) Design issues for cDNA microarray experiments. *Nat Rev Genet.*, **3**, 579-588.

Clustering Algorithms Optimizer: A Framework for Large Datasets

Roy Varshavsky[1,*], David Horn[2], and Michal Linial[3]

[1] School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel
royke@cs.huji.ac.il
[2] School of Physics and Astronomy, Tel Aviv University, Israel
[3] Deptartment of Biological Chemistry, Institute of Life Sciences, The Hebrew University of Jerusalem, Israel

Abstract. Clustering algorithms are employed in many bioinformatics tasks, including categorization of protein sequences and analysis of gene-expression data. Although these algorithms are routinely applied, many of them suffer from the following limitations: (i) relying on predetermined parameters tuning, such as a-priori knowledge regarding the number of clusters; (ii) involving nondeterministic procedures that yield inconsistent outcomes. Thus, a framework that addresses these shortcomings is desirable. We provide a data-driven framework that includes two interrelated steps. The first one is SVD-based dimension reduction and the second is an automated tuning of the algorithm's parameter(s). The dimension reduction step is efficiently adjusted for very large datasets. The optimal parameter setting is identified according to the internal evaluation criterion known as Bayesian Information Criterion (BIC). This framework can incorporate most clustering algorithms and improve their performance. In this study we illustrate the effectiveness of this platform by incorporating the standard K-Means and the Quantum Clustering algorithms. The implementations are applied to several gene-expression benchmarks with significant success.

Abbreviations and Keywords: Bayesian Information Criterion (BIC), Quantum Clustering (QC), Optimal K-Means (OKM), Optimal Quantum Clustering (OQC), Principal Component Analysis (PCA), Singular Value Decomposition (SVD).

1 Introduction[1]

In the field of genomics and proteomics, as well as in many other disciplines, categorization is a fundamental challenge. Categorization is defined as systematically arranging elements (data-points) into specific groups. Clustering, being an unsupervised learning problem, may be regarded as a special case of categorization with unknown

[*] Corresponding author.

[1] **Availability and Supplementary material:** The framework has been implemented in MATLAB (Version 6.5), and is freely available at http://adios.tau.ac.il/compact/framework

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 85–96, 2007.

labels (for further details see [1, 2]). Some algorithms such as CLICK [2], CTWC [3, 4] and CAST [5] were primarily developed for large sets of biological data while others were adopted from other fields (e.g., K-Means, Fuzzy C-means [6], Agglomerative Hierarchical Clustering, Self Organized Maps). One of the algorithms that we will expand on is Quantum Clustering (QC), the effectiveness of which has been demonstrated on gene-expression data [7, 8].

In large scale gene-expression tasks, clustering algorithms are useful for diagnosis of different samples (e.g., differentiating sick and healthy tissues, associating tissues with subtypes of a disease) as well as revealing functional classes of genes among the thousands often used in experimental settings [9].

Methods for collecting expression levels on a genome-wide level have been rapidly improving, leading to increased amounts of data to be analyzed. Additionally, much of the biological data is represented in high dimensions. Some clustering algorithms do not perform well when applied to large high-dimensional datasets. In particular, several model-based algorithms that are shown to be very efficient on limited size datasets [10], are found unfeasible when large scale datasets arc introduced (for computational complexity discussion see [11] and supplementary). The hope is that efficient preprocessing will address the task of computational feasibility while efficiently remove noise, thus allowing exposure of meaningful features of the data.

It would be presumptuous to propose one preprocessing protocol that works for all kinds of data. Different preprocessing methods are based on averaging and variance standardization, excluding genes with low variance between conditions [2], PCA, Fourier transforms [12], and more.

One fundamental preprocessing direction is dimension reduction. Ding *et al.* claim that the dimension should be correlated with the expected number of clusters [13]. However, this may not hold for real biological data, since this argument is based on a model in which data are generated by independent Gaussian distributions. Moreover, in many cases the number of clusters is unknown.

Several efforts to develop efficient and accurate filtering schemes and compression tools have been proposed [14, 15]. A routine scheme for gene-expression data (including commercial analysis tools provided by various platforms) is to filter elements in a supervised manner. For example, genes whose variance is below a certain threshold for different experimental conditions are discarded. Obviously, such filtering is often biased and misses a genuine property of the data.

In addition to preprocessing, clustering algorithms usually require selecting a set of parameters, thus turning each application into a set of subjective choices. If no prior knowledge is available, assessing the correct number of clusters (e.g., as required by the K-Means algorithm), is almost impossible. This choice is avoided by hierarchical algorithms that propose some O(N) possible partitions[2] of varying sizes, and the decision on the best partition is user determined.

Several of the most successful algorithms in the field of gene-expression do not explicitly accept the number of clusters K as an input; however this number is directly derived from their parameters. Amongst them are (*i*) the CAST algorithm [5], in which the affinity threshold determines the number of clusters, *(ii)* the CLICK

[2] In the paper N refers to the number of elements in the data, and K denotes the number of clusters.

algorithm [2], in which the homogeneity value determines K by controlling the kernels and the definition of singletons. *(iii)* The CTWC algorithm [4] where some parameters (such as stability threshold and minimal group size) determine K, and *(iv)* QC [7] where the Parzen window size (σ) determines the number of clusters.

Moreover, algorithms such as K-Means, Fuzzy C-Means and others, being nondeterministic, are inconsistent as they depend on starting points and other stochastic factors. Some methods such as averaging clustering results, following a majority rule, or applying other heuristics [16] have been suggested.

Since different results may be obtained by the numerous clustering algorithms that exist, evaluation of this variety is an essential step of the analysis [17, 18], and a reliable method is required. In this study we present a framework to overcome the pitfalls described above by (i) a generic method for preprocessing and (ii) a measure based on an internal criterion that can be incorporated in any clustering algorithm.

2 Methods

Our proposed framework includes two interrelated steps: preprocessing and parameter tuning. We outline the rationale of the method and describe its implementation on two different kinds of clustering algorithms.

2.1 Preprocessing

Singular Value Decomposition (SVD) serves as a good and efficient preprocessing step and is useful for dimension reduction [8, 12, 19].

SVD represents any real matrix X as a product $X=U\Sigma V^T$, where U and V are orthonormal matrices and Σ is a diagonal matrix whose eigenvalues s_i (singular values) appear in decreasing order. The columns of U and V define two independent vector spaces. This decomposition is unique (up to overall phases) and holds for any real matrix of size m by n. The number of non-zero entries in Σ equals the rank of X. A common application of SVD is dimension reduction: this is performed by replacing Σ with a truncated version where only a small number (r) of leading singular values is retained and the rest are replaced by zeros. The resulting reconstructed matrix X' ($X'=U\Sigma'V^T$), is the best least-mean-squares approximation of X obtainable by any matrix of rank r.

We focus our attention on the matrices U and V. In a problem where X is a matrix of m genes by n samples, U and V form representations of gene and sample spaces respectively. It is within these spaces, now reduced to rank r that we look for cluster structures [8].

How does one choose the rank r of the truncated space? The singular values s_i have the meaning of standard deviations. Defining the relative variance V_i of component i (see Fig 1A and supplementary), one may come up with several principles for truncation.

$$V_i = s_i^2 / \sum_{j=1}^{N} s_j^2 \tag{1}$$

Wall [12] suggested the following guidelines: (1) ignore components beyond the point where the cumulative relative variance becomes larger than a certain threshold (e.g. 85%), (2) ignore components with relative variance below a certain threshold (e.g. 1%), or (3) stop when a sudden decrease is observed in the relative variance graph. We suggest using SVD- entropy [19] as a guide for choosing among the possibilities.

$$E(Data) = -\frac{1}{\log(N)} \sum_{i=1}^{N} V_i \log(V_i) \tag{2}$$

E varies between 0 and 1. *E* = 0 corresponds to an ultra ordered dataset that can be explained by a single eigenvector (problem of rank 1) and *E* = 1 stands for a disordered matrix in which the spectrum is uniformly distributed. We find that in gene-expression datasets, entropy values are higher than 0.5, reflecting a disordered distribution. If *E* is very low, a sudden decrease in the spectrum is a good indicator for the best *r* values. Otherwise we prefer criteria *(1)* and *(2).*

Truncation to dimension *r* is equivalent to projecting the vectors of our problem (e.g. the genes or samples vectors) onto an r-dimensional subspace. The vectors, as defined in this subspace, have different norms. It is preferable to renormalize the vectors, i.e. project them onto the unit hyper-sphere in r-space. This approach considers similarity between vectors in the truncated space in terms of the cosine of the angle between them, and is consistent with the standard application of Latent Semantic Analysis (LSA) [20]. It is worth mentioning that, although we suggest using SVD, other truncation methods may be used (e.g., Fourier transforms, PCA).

2.2 Parameter Tuning

The validity and reliability of clustering algorithms may be questioned on two grounds: *(1)* subjectivity, i.e. using supervised criteria in the parameter setting and *(2)* inconsistency, i.e. obtaining different results upon repeated application of nondeterministic algorithms.

In order to reduce these pitfalls to a minimum, we suggest using an internal criterion. The criterion we choose to adopt is the Bayesian Information Criterion (BIC). Fraley and Raftery [21] developed it in a model-based analysis that assumed the data to be generated by a mixture of underlying normal probability distributions. The parameters of the underlying distributions were set by an EM algorithm. The BIC criterion is used to evaluate the number of clusters and the quality of the suggested clustering. BIC is defined as follows:

$$BIC \equiv 2l_M(x,\hat{\Theta}) - m_M \log(N) \approx 2\log p(x \mid M) + const \tag{3}$$

where $l_M(x,\Theta)$ is the mixture log likelihood (of the data *x* and the predicted model Θ), which is maximized under the constraint that m_M (a function of the number of independent parameters[3]), is minimized. It is assumed that a higher BIC score reflects better clustering quality. Recently, Teschendorff *et al.* have applied an EM algorithm to find a partition that maximizes the BIC criterion [10]. Here we do not optimize the

[3] We choose m_M*=dim*K* (K+dim),* where dim is the number of dimensions and K is the number of clusters.

BIC score. Trusting the clustering algorithms we just use this score, in a way befitting the algorithms, to find the best clustering parameters.

3 Implementation

We demonstrate our method on two fundamentally different clustering algorithms. They differ in some fundamental aspects thus testing the generality of our framework.

Optimized K-Means (OKM)

K-Means is a very popular, fast and intuitive algorithm. This naïve algorithm has two known drawbacks: First, it requires the number of clusters as an input, and thus is limited to scenarios where external knowledge is available. Secondly, the algorithm is nondeterministic, and is thus inconsistent.

The OKM implementation applies the K-Means algorithm 50 times for each number of clusters (K=1 to 20 in our examples) and computes the BIC score for each application. The application that leads to the maximal BIC score is considered to be the optimal solution.

Optimized QC (OQC)

The QC algorithm [7] uses the Schrödinger equation to provide an effective clustering description of the data. It requires one parameter, σ, a Parzen window width. This parameter controls the number of clusters that are identified by the algorithm with larger values of σ yielding fewer clusters. Different σ may also yield the same number of clusters but different clustering assignments (see Fig. 2B). Contrary to K-Means this algorithm is deterministic, has less constraints than K-means (since noise is integrated within the model), and does not assume spherical properties of the clusters. Recently, a variation of the algorithm's convergence, using the mean-shift approach, was suggested [22]. Here we employ the standard implementation [7].

OQC consists of applying QC once for a set of σ values (50 values in the range of 0.1 to 0.9, in our examples), and computes the BIC score for each σ. The maximal BIC is considered as the optimal solution.

4 Results

Here we describe our results on three gene-expression datasets that are well known benchmarks. In the first [23] and the second [24] examples, samples were clustered (2 and 4 clusters, respectively) while in the third dataset [25] clustering was performed on the genes. All three cases have assignments that were manually curated. The assignments serve to estimate the performance of the clustering algorithms, using the Jaccard score which reflects the ‘intersection over union' between the algorithm's clustering assignments and the expected classification[4]:

$$Jaccard = \frac{n_{11}}{n_{11} + n_{01} + n_{10}} \tag{4}$$

[4] We refer to supplementary material for further explanation.

4.1 The Colon Dataset of Alon et al. (1999)

In the dataset of [23], 62 gene-expression samples were taken from colon cancer patients. 40 of them were taken from sick tissues, and 22 from healthy tissues. Each sample contains the expression of 7479 genes. We follow [23, 24] who chose 2000 genes with the highest confidence in the measured expression levels.

In order to emphasize the influence of preprocessing on the clustering results, we compare SVD (see methods) with Principal Components Analysis (PCA)[5]. Fig 1A displays the singular values of the [2000x62] matrix.

The compression guidelines (see methods), suggests that only 2 or 3 components may be needed for a good description of the data (the relatively low entropy: 0.28, see equation 2). This yields compression rates of $1x10^{-3}$ and $1.5x10^{-3}$, respectively.

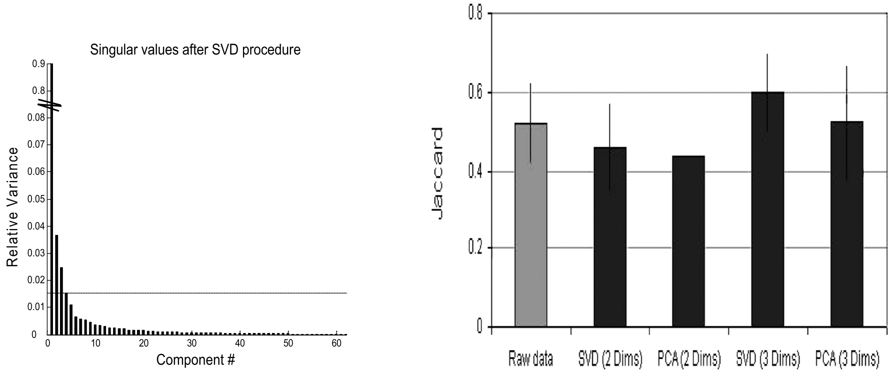


Fig. 1. A. (left) Singular values of the colon dataset (dashed line denotes the 'cut' decision). **B.** (right) Jaccard scores of the KM on raw data (left bar) and different preprocessing options.

As shown in Fig. 1A, preprocessing procedure influences the clustering quality. We conclude that this step deserves substantial attention. Moreover, when selecting the correct compression method (SVD in 3 dimensions), the clustering results are improved, as reflected by the increase in the Jaccard score (from 0.52 to 0.6).

The optimal results are obtained for SVD reduction to 3 dimensions. At this stage, the data are compressed to 62 vectors on a 3 dimensional unit sphere. Fig. 2A displays the OKM results (50 executions for 2-20 putative clusters) for different choices of K. For each K the maximal BIC of all 50 trials was chosen. The overall maximal BIC value is obtained for K=2. Note that the farther the number of clusters is from the correct solution, the larger is the dispersion of the corresponding BIC values. Comparing the internal (BIC) and external (Jaccard) criteria, one finds that the K=2 assignments were also the closest to the experts opinion. This testifies to the usefulness of BIC as an indicator of the proper clustering of the data.

[5] Matlab code: `princomp(zscore(X'X))`.

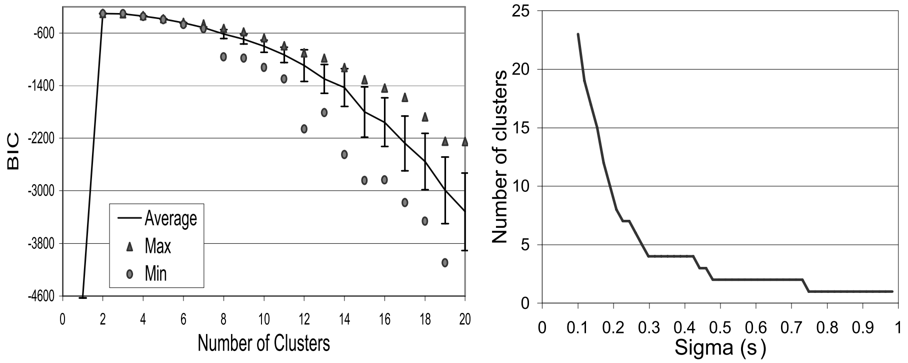


Fig. 2. A. (left) BIC Values when applying OKM (SVD reduced to 3 dimensions) on the colon dataset. **B.** (right) The number of clusters obtained in the colon dataset as a function of the σ input parameter of the QC algorithm.

Next we apply OQC to the compressed colon dataset. Recall that QC is a deterministic algorithm, thus, a single application is required for each σ value. Fig. 2B displays the number of clusters when varying σ. Note that different σ values may lead to the same number of clusters but different assignments, hence BIC may vary when the number of clusters remains constant.

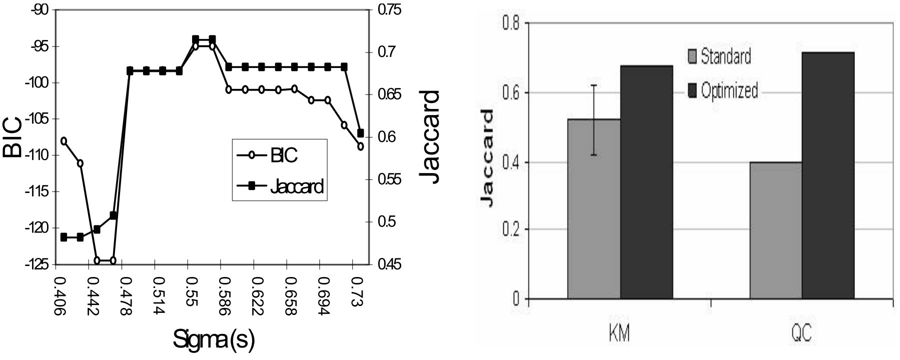


Fig. 3. A. (left) Comparison of the internal (BIC) and external (Jaccard) criteria for the colon dataset (OQC). **B.** (right) Comparison of the standard and optimized versions of the KM and QC algorithms.

Both BIC and Jaccard scores display the same behavior in the neighborhood of their maximal values (Fig. 3A). The maximal BIC was obtained for σ=0.55, where QC leads to 2 clusters. The corresponding Jaccard score for this σ is 0.715.

Since both OKM and OQC share the same preprocessing step, their clustering results can be compared. The maximal BIC value achieved by OQC is higher than the one achieved by OKM (-95 and -300, respectively). Similarly, the Jaccard score of the

OQC is higher than the one of OKM (0.715 and 0.678, respectively). Fig. 3B compares these results with what the same algorithms obtain on the original datasets without preprocessing (0.52 and 0.4 for KM and QC, respectively). The results are even more impressive when compared to other state-of-the-art algorithms (Table1).

Table 1. Jaccard scores of various algorithms when applied to the Alon dataset

Method	Jaccard
K-Means (raw data, 50 repeats)	0.52 (0.1)
OKM (Preprocessing & BIC)	0.678
QC (raw data)	0.4
OQC (Preprocessing & BIC)	**0.715**
CLICK [2]	0.64
CAST [2,5]	0.682
CTWC ([4], and[6])	0.508

4.2 The Leukemia Dataset of Golub et al., 1999

The dataset of Golub *et al.* has served as a benchmark for several clustering methods [2, 4 and 24]. The experiment sampled 72 leukemia patients with two types of leukemia, ALL and AML. The ALL set is further divided into T-cell leukemia and B-cell leukemia and the AML set is divided into patients who have undergone treatment and those who did not. For each patient, an Affymetrix GeneChip measured the expression of 7129 genes. The clustering task is to find the four cancer groups within the 72 patients in a [7129x72] gene expression matrix. We select the first five eigenvectors, achieving a compression rate of $7x10^{-4}$ (from [7129x72] to [5x72]).

BIC is maximized for K=2 in OKM, as is the Jaccard score (Fig. 4A). Hence we conclude that OKM can identify only the two major groups in the data and cannot detect a partition into four groups. This finding is consistent with the CAST and CLICK algorithms that have also failed to identify the subtypes [2]

Since QC cannot be applied to the raw dataset, preprocessing is of essence. OQC proves to be very effective. As displayed in Fig. 4B, the correlation between the BIC and the Jaccard scores is quite high around the maximum of both curves. Moreover, the maximum BIC is at σ =0.548, which dictates partitioning into 4 clusters, similar to what would be expected from the data. The corresponding Jaccard score for this σ is 0.69 (Fig. 4B). 4 clusters are predicted by QC throughout the range $0.47<\sigma<0.56$.

4.3 The Yeast Dataset of Spellman et al. (1998)

The dataset of [25] presents a somewhat more challenging task than the previous examples, since we examine our method on clustering of genes. Spellman *et al.* identified 798 genes as cell cycle regulated and assigned them to 5 different stages of the yeast cell cycle (M/G1, G1, S, G2 and M). Expression levels of these genes were recorded at 72 time points, yielding a [798x72] matrix.

[6] http://www.weizmann.ac.il/physics/complex/compphys/ctwc/

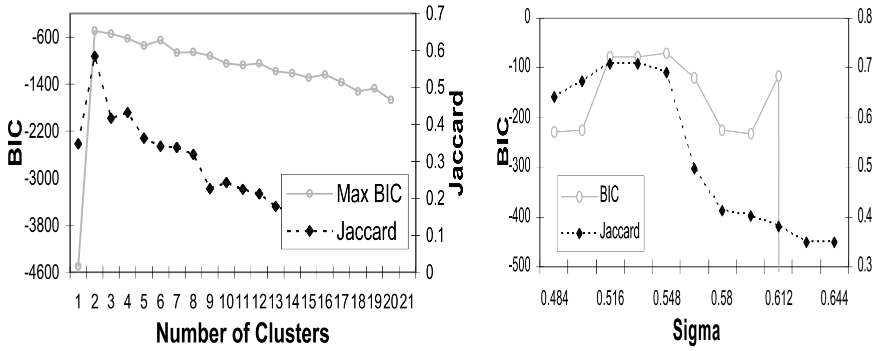


Fig. 4. A. (left) BIC and Jaccard scores of the Golub dataset (OKM), **B.** (right)Comparison of internal (BIC) and external (Jaccard) criteria of the leukemia dataset (OQC)

Contrary to the first examples, the distribution of relative variances is gradual and the entropy is significantly higher (0.705, see supplement). This result is consistent with the argument that high entropy reflects data that were preprocessed, since genes were intentionally selected by their functional annotation. We selected the first four leading eigenvectors (note the dashed line in the figure) achieving a compression rate of $5x10^{-2}$ (from [798x72] to [798x4]).

The external expert [25] suggests that there are 5 groups of cell cycle related genes. When applying the OKM protocol to the compressed dataset a maximized BIC is observed at 6 clusters. Comparing to the standard application of K-Means, the OKM shows no improvement: both applications yield Jaccard scores of 0.4.

Application of OQC to the compressed dataset yields a somewhat different result than that of OKM. BIC is maximized at σ=0.5, where 4 clusters are identified. Taking a closer look at the OQC clusters suggests that the S and G2 stages are joined by QC into one cluster. Here the correlation between the BIC and Jaccard scores is not perfect (see supplementary). Nevertheless, the Jaccard score it yields is relatively high (0.5 comparing to 0.4 in many other algorithms, see supplement table).

5 Conclusion

We present a general ‘clustering improver’ scheme. This unsupervised, data-driven two-step clustering framework uses intrinsic properties of the dataset to determine the SVD-based compression. After dimension reduction, several iterations of a clustering algorithm are applied, each with a different parameter. They are then compared with each other by the BIC criterion. The parameter that yields the best BIC score is chosen and is declared to be the optimal one. This generic framework is also computationally efficient: it processes these large-scale datasets on a standard PC in less than a minute (e.g., 50 runs of each of the different number of clusters in OKM).

Preprocessing of experimental data is an essential step. The raw data often come in a large-scale, un-normalized and noisy representation. These distractions have to be treated. Nevertheless, due to the diversity of the experiments one cannot provide a universal preprocessing method. In our study, we emphasize the importance of compression, and present some examples of the variations that different preprocessing methods can yield. We recommend SVD-based compression, which provides a normalized, filtered and ultra-compressed representation of the data. We also suggest guidelines regarding the extent of the compression.

The second step of our methodology is parameter tuning, which is based on the BIC score. Choosing this score has two advantages: *(1)* being an internal measurement, it allows an unbiased, automated method with no external intervention, and *(2)* its capability to be computed after the algorithm has terminated its application allows this independent criterion to be ‘plugged in’ to any clustering algorithm.

BIC is useful for finding the best solution amongst many local maxima, for both deterministic and nondeterministic clustering algorithms. Some heuristics are proposed in order to overcome the inconsistency problem of nondeterministic algorithms. In cases where many applications of the same algorithm lead to suboptimal solutions and only a few suggest good solutions, BIC maximization represents considerable improvement over other methods such as majority voting. Even if BIC does not point to the best clustering solution, it chooses one that is close to the best. It can therefore assist in narrowing down the search for best parameters.

Our methodology is especially well adapted to algorithms that assume spherical distribution (e.g., K-Means) of clusters, but it can be applied to algorithms that do not assume such a distribution. Surprisingly, it performs very well for methods that do not subsume spherical clustering such as QC and SOM (not shown). The optimized algorithms described here outperform the published results of CTWC, CLICK and CAST. We assume the same methodology to the latter algorithms could improve their performance even further.

Nevertheless, we identify some limitations. First, as we have not suggested any modification in any clustering algorithm per se, the improvement is bounded to the algorithm’s best performance. If the solution space does not describe the underlying structure of the dataset, we cannot obtain a high quality solution.

Second, the BIC score assumes a specific hyper-elliptic organization of clusters. When, as in the yeast dataset, clusters have different distributions, BIC has less descriptive strength. In such cases BIC may not fit the properties of the dataset. Third, the BIC value, computed by the EM method, usually cannot converge when the number of dimensions surpasses some threshold (of the order of 10). An efficient preprocessing is therefore a prerequisite for the BIC to be computed.

Finally, since BIC fits a model to a specific data distribution, it cannot be used to compare models of different datasets. For the same reasons it cannot be used to choose among different preprocessing methods or truncated dimensions.

Different clustering algorithms are currently included in analysis suites that are applied by experimentalists to gene expression data. A standard practice is to apply several algorithms with a few configurations and choose among them on the basis of some known classification. Our framework may serve as a platform for systematic comparison between different clustering algorithms. In all comparisons, analysis is applied to an identical experimental benchmark. The large variation in performance of each algorithm supports the notion that there is no 'one-size-fits-all' method. This study attempts to reduce the subjectivity in data interpretation by providing a platform for comparisons that can be adopted by any algorithm.

Acknowledgements. We thank Noam Kaplan, Alon Kaufman and Menachem Fromer for critical reading the manuscript. R.V. has been awarded a fellowship by the Sudarsky Center for Computational Biology. This work is partially supported by EU Framework VI, DIAMONDS consortium.

References

1. Jain AK, Dubes RC: Algorithms for Clustering Data. Englewood Cliffs, NJ: Prentice Hall; 1988.
2. Sharan R, Shamir R: CLICK: A Clustering Algorithm with Applications to Gene Expression Analysis. In*: 2000*: AAAI Press, Menlo Park, CA; 2000: 307–316.
3. Blatt M, Wiseman S, Domany E: Superparamagnetic Clustering of Data. *Physical Review Letters* 1996, 76:3251–3254.
4. Getz G, Levine E, Domany E: Coupled two-way clustering analysis of gene microarray data. *PNAS* 2000, 97(22):12079-12084.
5. Ben-Dor A, Shamir R, Yakhini Z: Clustering Gene Expression Patterns. *Journal of Computational Biology* 1999, 6(3-4):281-297.
6. Dembele D, Kastner P: Fuzzy C-means method for clustering microarray data. *Bioinformatics* 2003, 19(8):973-980.
7. Horn D, Gottlieb A: Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Physical Review Letters* 2002, 88(1).
8. Horn D, Axel I: Novel clustering algorithm for microarray expression data in a truncated SVD space. *Bioinformatics* 2003, 19(9):1110-1115.
9. Eisen MB, Spellman PT, Brown PO, Botstein D: Cluster analysis and display of genome-wide expression patterns. *PNAS* 1998, 95(25):14863-14868.
10. Teschendorff AE, Wang Y, Barbosa-Morais NL, Brenton JD, Caldas C: A variational Bayesian mixture modelling framework for cluster analysis of gene-expression data. *Bioinformatics* 2005, 21(13):3025-3033.
11. Zhong S, Ghosh J: A unified framework for model-based clustering. *Journal of Machine Learning Research* 2003, 4(964287):1001-1037.
12. Wall M, Rechtsteiner A, Rocha L: Singular Value Decomposition and Principal Component Analysis. In: *A Practical Approach to Microarray Data Analysis.* Edited by Berrar D, Dubitzky W, Granzow M: Kluwer; 2003: 91-109.
13. Ding C, He X, Zha H, Simon H: Adaptive dimension reduction for clustering high dimensional data. In: *IEEE International Conference on Data Mining: 2002*; 2002: 107-114.
14. Xing EP, Karp RM: CLIFF: clustering of high-dimensional microarray data via iterative feature filtering using normalized cuts. *Bioinformatics* 2001, 17(90001):S306-315.
15. Plagianakos VP, Tasoulis DK, M.N. V: Hybrid dimension reduction approach for gene expression data classification. In: *International Joint Conference on Neural Networks 2005, Post-Conference Workshop on Computational Intelligence Approaches for the Analysis of Bioinformatics: 2005.*
16. Zhong W, Altun G, Harrison R, Tai PC, Pan Y: Improved K-means Clustering Algorithm for Exploring Local Protein Sequence Motifs Representing Common Structural Property. In: *IEEE Transactions on NanoBioscience: 2005*; 2005: 255-265.
17. Handl J, Knowles J, Kell DB: Computational cluster validation in post-genomic data analysis. *Bioinformatics* 2005, 21(15):3201-3212.

18. Varshavsky R, Linial M, Horn D: COMPACT: A Comparative Package for Clustering Assessment. In: *Lecture Notes in Computer Science.* 3759 ed: Springer-Verlag; 2005: 159-167.
19. Alter O, Brown PO, Botstein D: Singular value decomposition for genome-wide expression data processing and modeling. *PNAS* 2000, 97(18):10101-10106.
20. Landauer TK, Foltz P. W., Laham D: Introduction to Latent Semantic Analysis. *Discourse Processes* 1998, 25:259-284.
21. Fraley C, Raftery AE: How many clusters? Which clustering method? - Answers via Model-Based Cluster Analysis. In: *Computer Journal.* vol. 41; 1998: 578-588.
22. Barash, D. and D. Comaniciu. *Meanshift clustering for DNA microarray analysis*. In Computational Systems Bioinformatics Conference (CSB) 2004: IEEE.
23. Alon U, Barkai N, Notterman DA, Gish K, Ybarra S, Mack D, Levine AJ: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *PNAS* 1999, 96(12):6745-6750.
24. Golub TR, Slonim DK, Tamayo P, Huard C, Gaasenbeek M, Mesirov JP, Coller H, Loh ML, Downing JR, Caligiuri MA *et al*: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. *Science* 1999, 286(5439):531-537.
25. Spellman PT, Sherlock G, Zhang MQ, Iyer VR, Anders K, Eisen MB, Brown PO, Botstein D, Futcher B: Comprehensive Identification of Cell Cycle-regulated Genes of the Yeast Saccharomyces cerevisiae by Microarray Hybridization. *Mol Biol Cell* 1998, 9(12):3273-3297.

Ranking Function Based on Higher Order Statistics (RF-HOS) for Two-Sample Microarray Experiments

Jahangheer Shaik and Mohammed Yeasin

Computer vision, pattern and image analysis lab(www.cvpia.org)
Electrical and Computer Engineering
University of Memphis
Memphis TN-38152
jshaik@memphis.edu, myeasin@memphis.edu

Abstract. This paper proposes a novel ranking function, called RFHOS by incorporating higher order cumulants into the ranking function for finding differentially expressed genes. Traditional ranking functions assume a data distribution (e.g., Normal) and use only first two cumulants for statistical significance analysis. Ranking functions based on second order statistics are often inadequate in ranking small sampled data (e.g., Microarray data). Also, relatively small number of samples in the data makes it hard to estimate the parameters accurately causing inaccuracies in ranking of the genes. The proposed ranking function is based on higher order statistics (RFHOS) that account for both the amplitude and the phase information by incorporating the HOS. The incorporation of HOS deviates from implicit symmetry assumed for Gaussian distribution. In this paper the performance of the RFHOS is compared against other well known ranking functions designed for ranking the genes in two sample microarray experiments.

Keywords: Two-sample microarray data, Higher order statistics, Differentially expressed genes.

1 Introduction

DNA microarrays became one of the most popular biotechnologies that allow the monitoring of expression levels of thousands of genes simultaneously. Several different platforms (for example, Affymetrix [1], Agilent [2] etc.) have been developed to assist researchers understand the difference in expression of genes in different cells and tissues. Microarray experiments produce expression profiles measured under some experimental conditions and are normally labeled on the basis of external information such as, clinical identification of tissue samples or expression of genes with respect to time [3]. Accurate information about over-expression or under-expression of genes when comparing diseased tissues with normal tissues provides significant clues in understanding the mechanism of the disease. The ability to simultaneously profile the differential expression of large number of genes is the essential first step to succeed in the field of drug target discovery, molecular diagnostics, functional genomics and pharmacogenomics [4]. It is anticipated that these discoveries will lead to new therapeutic and diagnostic tools.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 97–108, 2007.

Despite all the possibilities and advantages in conducting large scale experiments, microarray technologies have some inherent problems, including dimensionality of the data samples, noise, and variabilities in the measurements. There are only small subset of genes that are considered to be relevant and the rest of the genes form noise and mask the underlying phenomenon. A number of computational problems arise when performing statistical significance analysis on two-sample microarray data. Some of the problems may include (but not limited to) [5, 6]: i) small sample size when compared to features; ii) inadequate understanding of the underlying model distribution and iii) experimental/technical noise.

One of the main objectives of knowledge discovery from microarray data is the selection of genes that are differentially expressed in different known classes of tissues. The state-of-the art methods for finding differentially expressed genes (DEGs) assume the data to follow a certain distribution (for eg. Normal). When the data characteristics deviate from the assumed distribution, the proposed methods do not yield expected results. Several studies have been performed in understanding and modeling the distribution of microarray data [7-13]. An inference may be drawn based on various studies that microarrays follow a wide variety of data distributions. Lonstedt and speed et. al. report the log normal distribution for the microarray data [11]. Purdom et al propose a second law of errors distribution (asymmetric Laplace distribution) which has heavy tails fits microarray data better than normal distribution [12]. Huber et al propose a skewed distribution based on 3 parameter lognormal distribution [9]. Chen et al propose a normal distribution for the microarray data [8]. Brody et al propose a lorentzian like distribution for the microarray data [7]. Several other references may be found to other model distributions.

A careful analysis of most of these data distributions reveals deviation from normality assumption. But, the ranking functions employed to find the DEGs in microarrays do not adequately consider these variations into account. This paper proposes a ranking function based on higher order statistics which consider the deviations from normality for ranking the marker genes from two sample microarray experiments. The higher order statistics considered are skewness and kurtosis. Skewness accounts for any non-symmetric properties of two-sample microarray data distribution. Kurtosis accounts for heavy or light tails of the data distribution. By incorporating higher order statistics it is possible to take care of these minor variations in the data. Another problem that is crucial while ranking the genes is the lack of large enough samples to estimate the parameters of the underlying distribution. As the number of samples increase, it is known that the data slowly approximates the normal distribution (c.f. central limit theorem [14]). Therefore, the number of samples plays an important role in determination of high ranked genes in the microarray data. This paper proposes inclusion of higher order statistics in the adaptive ranking function to address the aforementioned problems.

The paper is organized as follows. Section II presents the brief state-of-the art to provide the context of the work. The mathematical formulation behind ranking function based on higher order statistics (RFHOS) is presented in the Section III. The empirical analyses on artificial and real datasets are presented in the Section IV. Following this, discussions are presented in the Section V and finally, Section VI concludes the paper with a brief conclusion and future directions.

2 Research Context

A plethora of mathematical techniques have been developed for finding DEGs in microarray data [3, 15-17]. The performances of various methods for finding DEGs are hard to quantify and compare as they yield significantly different results on the same dataset. This problem can be attributed to the assumptions behind the currently used ranking functions as well as to the unique characteristics of microarray data. Also, very few empirical studies were conducted to compare the performance, and identify the shortcomings of such ranking functions in finding the DEGs [18, 19].

The early computational methods reported in the literature for finding DEGs are based on fold changes [20]. These methods, however, do not take into consideration the sample variance of the data. As shown in the Fig. 1, all the three possible cases have same mean but their similarity depends on the variance of the data which cannot be captured using fold change. For example, the two classes shown in the Fig. 1 (a) are very similar (high overlap), ones shown in Fig. 1(b) are moderately similar and the ones shown in the Fig. 1(c) are very dissimilar. They all have same mean but different variances. The higher the variances, more similar are the classes and vice versa. Any simple mean based approaches such as fold change method will rank all of them in the same category, which is very likely to be incorrect. The alternative is to use 2 sample t-test [21], which approximates the variance as a function of variances from two cases [22] as shown in Eq. 1. Since the microarray datasets have small number of samples, the sample variance may not be an indicative of actual variance. To address this problem, Thomas et al. [23] proposed a method that uses normalization constants for sample variances to obtain pooled variance. In [24], Tusher et al. have pointed out that small sample variance values yield false alarms for DEGs. They introduced an additive constant to the variance to reduce the false detection rate. This parameter estimation was proposed in [19] as 90th percentile of the sum of gene specific global standard errors. Mukherjee et al. [16] proposed the notion of reproducibility to minimize expected loss in determination of test statistics. Three parameters were introduced in the ranking function and were determined using Monte Carlo simulation. Recently, Shaik et al. [17] demonstrated that such ranking functions are susceptible to failure for a number of reasons. To improve the robustness a combined ranking method was proposed to achieve consistent ranking.

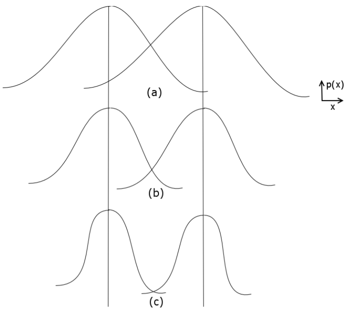


Fig. 1. Schematic diagram illustrating similarity between the Gaussian distributions with different variances

All the above mentioned methods inherently assume the distribution of the data to be Gaussian. However, for most practical situations this is not true. The Gaussian assumption implicitly implies the distribution is symmetric and may be approximated by 1st and 2nd order statistics which contain only amplitude information. However, it is not true for most practical cases. It may be seen that there is some skewness associated with the distributions. This skewness plays an important role in the determination of similarity as illustrated in the Fig. 2. The Fig. 2(a) shows the case where left probability density function (pdf) is positively skewed and right pdf is negatively skewed. This causes higher over lap among the pdfs and hence the two classes have highest similarity. On the contrary, Fig. 2(g) shows the case where left pdf is negatively skewed and right pdf is positively skewed. This causes low overlap among the pdfs resulting in lowest similarity. Rest of the cases in the Fig. 2 may be explained in similar manner. Please note that all of these cases as shown in Fig. 2 (a) to (h) have the same variance and mean estimates. This shows that the similarity/dis-similarity among the different sample cases cannot be captured using variance alone. The skewness plays an important role in the determination of (dis)similarity. In addition to that, the third order statistics contain the phase information of the signal [25]. The phase information coupled with the amplitude information may yield better results.

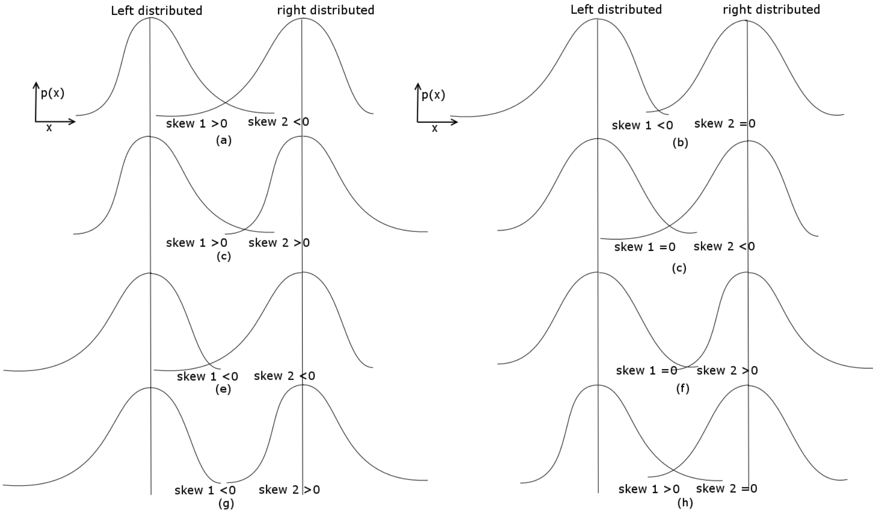


Fig. 2. Schematic diagram illustrating the impact of skewness in computing the (dis)similarity measures

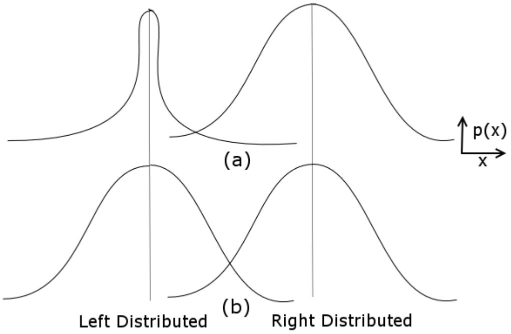


Fig. 3. Schematic diagram illustrating the impact of Kurtosis in computing the (dis)similarity measures

Kurtosis, commonly referred to as the fourth cumulant is a measure of peakness. It is also a measure of independence between the data distributions [26]. Higher the kurtosis, more independent (dissimilar) the distributions are and vice versa. It provides interesting insight into the variance estimate. The kurtosis of distribution implies that much of the variance is due to infrequent extreme deviations as opposed to frequent modest deviations. Fig. 3 shows the influence of kurtosis on the similarity which cannot be captured using the second order statistics alone. It can be inferred from the Fig. 3(a) that high kurtosis may mean less similarity (low over lap) and low kurtosis may mean high similarity (high over lap)as shown in Fig. 3(b).

3 Mathematical Formulation

Let ‘ $D_{nx(m+k)}$ ’ be the data matrix with ‘*n*’ genes (along the rows) and ‘*m*’ samples under one condition (say tumor class) and ‘*k*’ samples under the other condition (say non-tumor class). For a two sample microarray data, the sample variance is given by [27],

$$S = \frac{S_1^2}{m} + \frac{S_2^2}{k}, \tag{1}$$

where, S_1 is the standard deviation of samples in class 1 and S_2 is the standard deviation of samples in class 2.

The Adaptive ranking function is given by [16],

$$R\ (\theta_1, \theta_2, \theta_3) = \frac{d + \theta_1}{\theta_2 * S + \theta_3}. \tag{2}$$

Here, ‘*d*’ is the difference of means between the two sample distributions, ‘*S’* is the sample variance and θs are the parameters that scale the second order statistics.

The Second order statistics as shown in Eq. 2 do not perform adequately for the skewed and heavy tailed distributions. The higher ordered statistics such as skewness and kurtosis are incorporated into the ranking function to make the ranking more robust.

Let ‘Sk_1’ be the skewness of the left distributed data and let ‘Sk_2’ be the skewness of the right distributed data then the total skewness is given by,

$$Sk = \pm Sk_1 \pm Sk_2\,. \tag{3}$$

If the left distributed data is negatively skewed then $Sk_1 = -\,Sk_1$, if the data is positively skewed then $Sk_1 = +\,Sk_1$ and if the distribution is not skewed then Sk_1=0. If the right distributed data is negatively skewed then $Sk_2 = +\,Sk_2$, if the data is positively skewed then $Sk_2 = -\,Sk_2$ and if the distribution is not skewed then Sk_2 =0.

Let ‘ ku_1 ’ be the kurtosis of the left distributed data, let ku_2 be the kurtosis of the right distributed data then total kurtosis is given by,

$$ku = \frac{1}{ku_1} + \frac{1}{ku_2} \qquad (ku_1, ku_2 \neq 0). \tag{4}$$

If ku_1 =0, $ku = \frac{1}{ku_2}$, if ku_2 =0, $ku = \frac{1}{ku_1}$ and if $ku_1 = ku_2$ =0 then ku =0,

i.e., to say higher the kurtosis, less is the overlap and hence higher is the dis-similarity between the two sample distributions.

The higher order statistics are incorporated into the ranking function as given by Eq. 5

$$R_p = \frac{d+\theta_1}{(S+\theta_4 Sk+\theta_5 Ku)\theta_2+\theta_3}. \tag{5}$$

Here, θs are the parameters that scale the sample statistics incorporated into the ranking function and '*d*' is the difference of means between the two sample distributions.

The values '*Sk*' and '*Ku*' supplement the value of '*S*'as shown in Eq. 5. The bootstrapping procedure from [27] is employed on the original dataset $D_{nx(m+k)}$ and *j* < *min(m,k)* samples are randomly selected from both cases and pooled to form the data '$D1_{nx2j}$' ('*j*' samples from each condition). The process is repeated to construct another dataset '$D2_{nx2j}$'. The ranking function of the Eq. 5 is applied independently on these two datasets to obtain the scores of the marker genes describing their differential expression. These scores are ranked and sorted from highest to lowest resulting in R_1 and R_2 in Eq. 6. Since these two datasets are the subset of the original dataset '$D_{nx(m+k)}$', they must produce similar ranking. The optimized set of parameters which result in high consistency (Eq. 7) between the rankings is obtained using Monte Carlo simulation [27]. Since it is adequate to test the consistency using a few high ranked genes '*h*', *h* =100 is employed in this paper. The first '*h*' high ranked genes are obtained from these two rankings resulting in two sets given by,

$$S_1 = R_1(1:h) \text{ and } S_2 = R_2(1:h) \tag{6}$$

and the consistency between the rankings is obtained by comparing these two sets is given by,

$$C = S_1 \cap S_2. \tag{7}$$

Please note that '*h*' in Eq. 6 is not indicative of number of significantly DEGs expected from the algorithm. The ranking function proposed in Eq. 5 is expected to provide better estimate of rankings as it incorporates the skewness and kurtosis aspects into the ranking function. The effectiveness of the proposed ranking function is studied in this paper for different distributions by comparing it with other well known ranking functions. The affect of varying sample size on the ranking function is also studied.

Table1. Parameters used for generating artificial microarray datasets

		Normal tissues (condition1)	Abnormal tissues (condition 2)
Non-DEGs	mean	0	0
	variance	Gamma distribution with mean 2, variance 2	
DEGs	mean	0	Normal distribution mean 3, variance 1
	variance	Gamma distribution with mean 2, variance 2	Gamma distribution with mean 2, variance 2

4 Empirical Analyses

The efficacy of RF-HOS is illustrated using several artificial microarray datasets following two different distributions and two real microarray datasets viz. Gastric cancer and leukemia datasets [28, 29]. The performance of RFHOS is compared with other well known ranking functions. The influence of sample size on the performance of ranking functions is first studied using a lognormal model. Next, the performance of well known ranking functions is studied using an asymmetric Laplace distribution for sample size of 12. Further, the RF-HOS is applied on two real microarray datasets to find the DEGs. The validation of these results is performed by comparing with the results of the author.

A. Simulated Case Study 1. Lognormal Model for Artificial Microarray Datasets

A lognormal distribution model is used to generate artificial microarray datasets as proposed by [11]. The artificial microarray datasets are generated based on a multivariate lognormal model. For non-DEGs, the means under both conditions are set to zero. For DEGs, the means under one condition are set to zero and for other condition are drawn from normal with mean 3 and variance 1. Unequal variances following a gamma distribution are used for DEGs as proposed in [11, 16]. The parameters used for generating the artificial microarray datasets are shown in table 1. The artificial dataset is created to have 2050 genes with ‘s’ samples under each of the two conditions. Here, ‘s’ = 5, 8, 12, 15, 18 and 20 samples respectively for each case as shown in Fig. 5. The first 50 genes are rendered differentially expressed and the rest 2000 are rendered non-DEGs. This process enables class labels for genes (DEGs or non-DEGs) for each generated artificial microarray dataset which can be used as ground truth to verify the outputs of the different algorithms used in this study. To obtain the significantly DEGs, the following procedure is

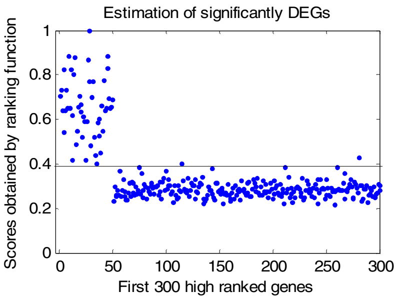
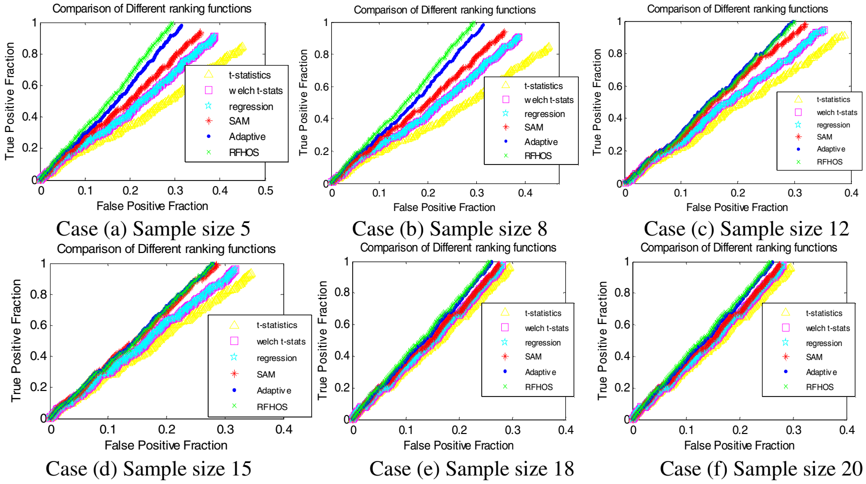


Fig. 4. Plot illustrating number of DEGs to be considered



Case (a) Sample size 5

Case (b) Sample size 8

Case (c) Sample size 12

Case (d) Sample size 15

Case (e) Sample size 18

Case (f) Sample size 20

Fig. 5. Reciever Operating Characteristic (ROC) Curves Showing the Performance of Different Ranking Algorithms for artificially generated data set 2

applied. The ranking function of Eq. (5) is first employed to obtain the scores of marker genes indicative of their differential expression. The procedure is repeated 25 times and the cumulative sum of scores for each gene is obtained. The scores of genes are normalized and plotted as a 2D map. The Fig. 4 shows the result obtained for one artificially generated dataset. The plot shows the normalized cumulative scores of first 300 high scored genes (out of 2050 genes). Since it is artificially generated data, it is known that first 50 are highly differentially expressed which is reflected in the plot. The estimate of significantly DEGs may be obtained as shown in Fig. 4 by drawing a decision boundary even without any prior knowledge of the number of DEGs.

The performance of different algorithms on set 1 for varying sample size is shown in Fig. 5. The true positive fraction (TPF) is obtained by finding the number of DEGs estimated by the ranking algorithm which are consistent with the ground truth and false positive fraction (FPF) is calculated by finding the number of DEGs not consistent with the ground truth. As shown in Fig. 5, RFHOS outperforms other algorithms in determining DEGs from the microarray data. For low sample size as shown by Figs. 5 case (a) and case (b), RFHOS and Adaptive ranking perform better than other ranking algorithms. Please note that performance of RFHOS is better than that of Adaptive ranking algorithm. As the sample size increases, the performance of different algorithms becomes similar as shown by the Figs. 5 case (e) and case (f). This may be because as the number of samples increase, the data tends to be more and more Gaussian as given by central limit theorem [14] and hence the higher order moments tend to become insignificant. It is evident from Fig. 5 that RFHOS performs better compared to other ranking algorithms used in the analyses.

B. Simulated Case Study 2: Asymmetric Laplace Distribution Model for Artificial Microarray Datasets

The artificial microarray datasets are created using the same procedure employed for lognormal distribution but by using an Asymmetric Laplace distribution as proposed by [12]. The mean and variance of the DEGs and Non-DEGs are approximated similar to the distributions in table 1. The sample size was set to 12. The performance of various well known ranking functions is tested using this data. Fig. 6 shows the ROC curves obtained for asymmetric Laplace distribution. The TPF and FPF are estimated as described in case study 1. As shown in Fig. 6, the RF HOS outperformed other ranking functions in finding DEGs from microarray data.

C. Leukemia Dataset

Gene expressions of approximately 6817 genes are used to classify two types of acute Leukemia viz. acute lymphoid leukemia (ALL) and acute myeloid leukemia (AML). The data consists of 47 (38 B-cell and 9 T-cell) cases of ALL and 25 cases of AML. The data is divided into a training class containing 38 samples and a test class containing 34 samples of tissues. The class labels for training class are available from the author [29]. The pre-processing proposed by the author resulted in 3571 genes, the rest of the genes are considered insignificant and hence eliminated. The data is further separated into training and test classes. Using the training data, the RF-HOS algorithm is applied to identify the genes that maximally differentiate between the two classes (ALL and AML). The RFHOS algorithm identified 48 genes out of 3571 genes to be highly differentially expressed (c.f. Sec IV-A). The list of 48 genes may be found at the link http://jshaik.com/LeuRankRFHOS.pdf.

To validate the DEGs, often a heat map is used in the reported literature [30]. The heat map provides qualitative visual depiction of patterns and is rendered not useful for complex datasets with multiple classes (more than 2). In this paper, a 3D star coordinate projection (3D SCP)[31] algorithm is used for validation. Fig. 7 shows the visualization of 38 training samples using 48 DEGs as features. As shown in Fig. 7, the 48 dimensional data is represented in 3 dimensions using the 3D SCP algorithm. Fig. 7 (a) shows the results using 3D SCP for training cases. The ALL cases are shown as 'red dots' and AML cases are shown as 'green stars' (original class labels are used). As shown by Fig. 7, the different classes of tissues are projected to distinct locations in 3D space with clear boundary between them. The 48 DEGs found using unified algorithm may thus be visually validated to be differentially expressed for the training class. Next, the same set of DEGs are used to project the entire data (training samples + testing samples) as shown in Fig. 7(b). As shown in Fig. 7(b), majority of the tissues separated into two distinct classes. The visualization results as shown in Fig. 7(b) are consistent with the class labels from the author [29].

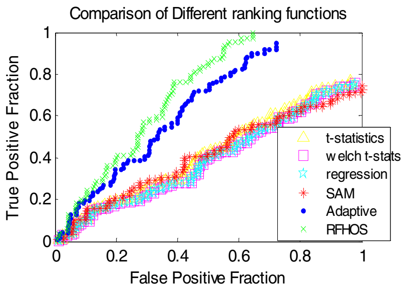
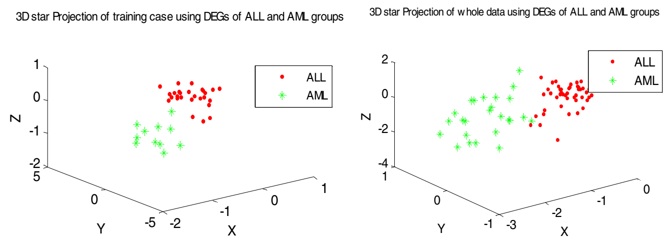


Fig. 6. Performance comparison of different ranking methods for Asymmetric Laplace distribution



(a) (b)

Fig. 7. 3D SCP projection of ALL and AML cases a) Training data and b) training and testing data

D. Gastric Cancer Dataset

The objective of this experiment is to identify genes distinguishing primary gastric cancers and metastatic gastric cancers from neoplastic gastric cancers which are otherwise morphologically indistinguishable. Approximately 30300 genes are used to study expression patterns of 90 primary gastric cancers, 14 metastatic gastric cancers and 22 neoplastic gastric cancers. The pre-processing steps mentioned in [28] are

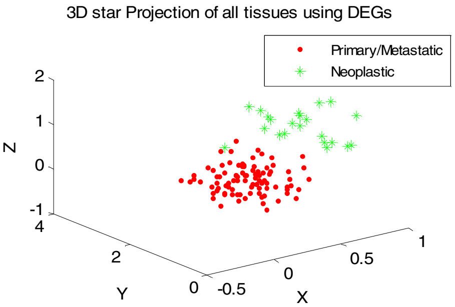


Fig. 8. 3D SCP of different tissues from Gastric cancer data

used resulting in 5200 genes for further study. The RFHOS algorithm is applied to find DEGs from neoplastic gastric cancers and other (primary/metastatic) gastric cancers. The ranked list of the genes may be accessed through the link http://jshaik.com/GastricRFHOScomm.pdf . Out of these genes, the first 82 were found to be significantly differentially expressed (c.f. sec IV-A). These 82 genes are compared with the list of genes found to be significant by the author [28]. All the 82 genes were present in the list of the genes found to be significant by the author.

The 3D SCP visualization algorithm [31] is used to see if the DEGs indeed separate different tissue cases. Fig. 8 shows the projection of the tissues (primary/metastatic Vs neoplastic) using the 82 significantly DEGs. The primary/metastatic tissues are represented by 'red dots' and neoplastic tissues are represented by 'green stars'. The tissues belonging to different classes are projected to different locations in 3D space as shown in Fig. 8 with clear boundary with them. The study using 3D SCP shows:

- The DEGs when used as features separated different tissue cases clearly as shown in Fig. 8.
- The metastatic cancers are indistinguishable from those of primary gastric cancers using these DEGs. This results concur with the initial observation by the author [28] that most of metastatic tumors arose from the primary tumors and hence are highly similar.

5 Discussions

The knowledge discovery from the microarray data consists of several objectives and one of the objectives is to find DEGs from the microarray data. Many methods are proposed to address this problem by taking into consideration the unique characteristics of the microarray data. Almost all of these methods assume the distribution to be Gaussian. It may be seen that by making such an assumption, some information is lost. It is argued that by incorporating higher order moments such as skewness and kurtosis, many problems in computing the similarity/dis-similarity may be addressed in a meaningful manner. These measures cannot be ignored by Gaussianity assumption because for Gaussian, higher order moments are zero. The proposed RFHOS incorporates the higher order cumulants into the ranking function. The moments are scaled by independent parameters estimated using Monte Carlo simulation (c.f. Sec- III). The performance of ranking functions for varying sample size is studied in this paper. It is seen from Fig. 5 (e and f) that as the number of samples increase, different ranking functions perform similarly. It is to be noted that as the number of samples increase, the data distribution approximates Gaussian and higher order moments tend to become zero. The application of RFHOS algorithm on Leukemia dataset [29] and Gastric cancer dataset [28] revealed potential genes that may be involved in carcinogenesis.

6 Conclusion

This paper presents a novel ranking function, called RFHOS that judiciously blends the higher order statistics into adaptive ranking function. Two well known models for microarray data viz. lognormal [11] and asymmetric Laplace distribution [12] are used to generate artificial microarray datasets. These artificial datasets with ground truth are used for the quantitative analysis of performance using the proposed RFHOS ranking function. To further illustrate the efficacy of the proposed algorithm empirical analyses were conducted on real microarray data to find the DEG's. In addition, the performance of the proposed algorithm is compared with other well known ranking algorithms. Empirical analyses on several datasets showed that the addition of higher order cumulants play an important role in determination of similarity/dis-similarity of the genes. It was observed that a RFHOS out performed other ranking functions considered for the analyses. The performance of ranking function for varying sample size is also evaluated using lognormal model of artificially generated microarray data. It is observed that as the sample size increases the performance of different ranking functions becomes similar. It is also seen that RFHOS performed better in finding DEGs from the real microarray data. The RFHOS ranking function is applied on Leukemia and Gastric cancer microarray datasets. The validation of the DEGs found is made in two steps. First, the DEGs are compared with the DEGs obtained by the authors[28, 29]. The 3D SCP algorithm is used next to visualize the tissues using DEGs obtained.

Acknowledgement. The authors acknowledge the Herff Fellowship and faculty start-up grants from the University of Memphis for partially funding of this research.

References

[1] P. A. Stephen, "Affymetrix, http://www.affymetrix.com/index.affx." Santa Clara, California, 1992-2007.

[2] B. Hewlett and D. Packard, "Agilent Technologies, http://www.home.agilent.com/agilent/home.jspx." Santa Clara, California, 1999-2007.

[3] I. Guyon, "An Introduction of Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157-1182, 2003.

[4] J. M. Ray and W. G. Hearl, "Methods for Evaluating Differential Gene Expression in Tissues and Cells," *Drug Development*, pp. 50-55, 2005.

[5] J. Shaik and M. Yeasin, "A Progressive Framework for Two-Way Clustering Using Adaptive Subspace Iteration for Functionally Classifying Genes," *Proceedings of IEEE IJCNN'06, Vancouver, Canada.*, pp. 5287-5292, 2006.

[6] J. Shaik and M. Yeasin, "Performance Evaluation of Subspace-based Algorithm in Selecting differentially Expressed Genes and Classification of Tissue Types from Microarray Data," *Proceedings of IEEE IJCNN'06, Vancouver, Canada.*, pp. 5279-5286, 2006.

[7] J. P. Brody, B. A. Williams, B. J. Wold, and S. R. Quake, "Significance and Statistical Errors in the Analysis of DNa microarray Data," *Proc Natl Acad Sci*, vol. 99, pp. 12975-12978, 2002.

[8] Y. Chen, E. R. Dougherty, and M. L. Bittner, "Ratio based decisions and quantitative analysis of cDNA microarray images," *Journal of Biomedical optics*, vol. 2, pp. 364-374, 1997.

[9] W. Huber, A. V. Heydebreck, H. Sultmann, A. Poustka, and M. Vingron, "Variance Stabilization Applied to Microarray Data Calibration and to Quantification of Differential Expression," *Bioinformatics*, vol. 18, pp. s96-104, 2002.

[10] T. Konishi, "Three Parameter Lognormal Distribution Ubiquitously Found in cDNA Microarray data and Its Application to Parametric Data Treatment," *Bioinformatics*, vol. 5, 2004.

[11] I. Lonnstedt and T. Speed, "Replicated Microarray Data," *Statistica Sinica*, vol. 12, pp. 31-46, 2002.

[12] E. Purdom and S. Holmes, "Error Distribution for Gene Expression Data," *Statistical Applications in Genetics and Molecular Biology*, vol. 4, 2005.

[13] D. M. Rocke and B. Durbin, "Approximate Variance-stabilizing Transformations for Gene Expression Microarray Data," *Bioinformatics*, vol. 19, pp. 966-972, 2003.

[14] R. O. Duda, P. E.Hart, and D. G.Stork, *Pattern Classification*, 2nd ed: John Wiley and Sons Inc, 2000.

[15] G. Getz, E. Levine, and E. Domany, "Coupled two-way clustering of gene microarray data," *Proceedings of National Academy of Science, USA*, vol. 97, pp. 12079-12084, 2000.

[16] S. Mukherjee, S. J. Roberts, and M. J. Laan, "Data-adaptive Test Statistics for Microarray Data," *Bioinformatics*, vol. 21, pp. 108-114, 2005.

[17] J. Shaik and M. Yeasin, "Adaptive Ranking and Selection of Differentially Expressed Genes from Microarray Data," *WSEAS transactions on Biology and Biomedicine*, vol. 3, pp. 125-133, 2006.

[18] W. Pan, "A Comparative Review of Statistical Methods for Discovering Differentially Expressed Genes in Replicated Microarray Experiments," *Bioinformatics*, vol. 18, pp. 546-554, 2002.

[19] I. B. Jeffery, D. G. Higgins, and A. C. Culhane, "Comparison and Evaluation of Methods for Generating Differentially Expressed Gene lists from MicroArray Data," *BMC Bioinformatics*, vol. 7, pp. 359-375, 2006.

[20] D. M. Mutch, A. Berger, R. Mansourian, A. Rytz, and M. A. Roberts, "The Limit Fold Change Model: A Practical Approach for Selecting Differentially Expressed Genes from Microarray Data," *BMC Bioinformatics*, vol. 21, pp. 3-17, 2002.

[21] H. Sahai and M. M. Ojeda, *Analysis of Variance for Random Models: Theory, Methods, Applications and Data Analysis*: Birkhauser, 2004.

[22] G. Casella and R. L. Berger, *Statistical Inference*, 2 ed: Duxbury Press, 2001.

[23] J. G. Thomas, J. M. Olson, S. J. Tapscott, and L. P. Zhao, "An Efficient and Robust Statistical Modeling Approach to Discover Differentially Expressed Genes using Genomic Expression Profiles," *Genome Research*, vol. 11, pp. 1227-1236, 2001.

[24] V. G. Tusher, R. Tibshirani, and G. Chu, "Significance Analysis of Microarrays Applied to The Ionizing Radiation Response," *PNAS*, vol. 98, pp. 5116-5121, 2001.

[25] A. Papoulis and S. U. Pillai, *Probability, Random Variables and Stochastic Processes*, 4 ed. New Delhi: Tata McGraw Hill, 2002.

[26] A. Hyvarinen and E. Oja, "Independent Component Analysis: Algorithms and Applications," *Neural Networks*, vol. 13, pp. 411-430, 2000.

[27] D. Stekel, *Microarray Bioinformatics*, 1 ed. Cambridge: Cambridge University Press, 2003.

[28] X. Chen, S. Y. Leung, S. T. Yeuen, K. M. Chu, J. Ji, R. Li, A. S. Y. Chan, S. Law, O. G. Troyanskaya, J. Wong, S. So, D. Botstein, and P. O. Brown, "Variation in Gene Expression Patterns in Human Gastric Cancers," *Mol Bio Cell*, vol. 14, pp. 3208-3215, 2003.

[29] T. R. Golub, D. K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J. P. Mesirov, H. Coller, M. L. Loh, J. R. Downing, M. A. Caligiuri, C. D. Bloomfield, and E. S. Lander, "Molecular classification of cancer: class discovery and class prediction by gene expression monitoring," *Science*, vol. 286, pp. 531-537, 1999.

[30] U. Alon, N. Barkai, D. A. Notterman, K.Gish, S. Ybarra, D. Mack, and A. J. Levine, "Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays," *Proc. Natl. Acad. Sci. USA*, vol. 96, pp. 6745-6750, 1999.

[31] J. Shaik and M. Yeasin, "Visualization of High Dimensional Data using an Automated 3D Star Co-ordinate System," *Proceedings of IEEE IJCNN'06, Vancouver, Canada.*, pp. 2318-2325, 2006.

Searching for Recombinant Donors in a Phylogenetic Network of Serial Samples

Patricia Buendia and Giri Narasimhan

Bioinformatics Research Group (BioRG), School of Computing and Information Science,
Florida International University, Miami, FL 33199, USA
{pbuen001,giri}@cis.fiu.edu

Abstract. Determining the evolutionary history of a sampled sequence can become quite complex when multiple recombination events are part of its past. With at least five new recombination detection methods published in the last year, the growing list of over 40 methods suggests that this field is generating a lot of interest. In previous studies comparing recombination detection methods, the evaluation procedures did not measure how many recombinant sequences, breakpoints and donors were correctly identified. In this paper we will present the algorithm *RecIdentify* that scans a phylogenetic network and uses its edge lengths and topology to identify the parental/donor sequences and breakpoint positions for each query sequence. *RecIdentify* findings can be used to evaluate the output of recombination detection programs. *RecIdentify* may also assist in understanding how network size and complexity may shape recombination signals in a set of DNA sequences. The results may prove useful in the phylogenetic study of serially-sampled viral data with recombination events.

1 Introduction

Modeling and detection of recombination is receiving increased attention, as is evidenced by the long list compiled at a website [1]. Recombination plays an important role in the evolution of genes and genomes; more importantly, it has a deleterious effects on the accuracy of phylogenetic reconstruction [2, 3]. Some programs only determine the presence or absence of recombination, without trying to infer recombination breakpoints [4]. Such a yes/no answer might be sufficient to decide which sequences to remove from a data set that will be used to infer a phylogenetic tree, thus justifying the evaluation process in many comparison studies of recombination detection methods [5-7]. The more sophisticated programs however, attempt to detect recombinant signals and donors for a "query" sequence [8-10].

Recombination has been largely ignored in the study of evolution due to the lack of practical methods that infer and reconstruct recombinant networks. Another reason is rooted in the belief that recombination may be disregarded when using certain genes, such as mitochondrial genes or genes from the Y-chromosome, which were thought not to recombine. However, a recent study has shaken these assumptions [11]. The study made use of the automated recombination detection methods of the package RDP2 [7, 10] to find evidence of the presence of recombination in mitochondrial DNA data sets used in published papers.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 109–120, 2007.

Detecting recombinants in a set of input sequences is a complex problem, and the performance of existing methods with regard to the accuracy of identification of recombinants, donors and breakpoint positions is not known. In this paper, we consider the simpler problem of identifying the recombinants when the input is a recombinant network of serially-sampled sequences. We show that this seemingly simpler problem is non-trivial. We point out that an efficient solution to such a problem will result in a tool that would be extremely useful for performing experiments with recombination detection.

In this paper, we discuss a new approach, *RecIdentify*, which seeks to determine the donor sequences and breakpoint positions of a set of serially-sampled sequences, whose relationships are specified by a given phylogenetic network. We discuss the conflicts that arise when such networks are large and contain multiple recombination events. The networks for our experiments were generated by the software Serial NetEvolve 1.0 [12], which attempts to emulate the evolution of rapidly recombining viruses such as HIV. The results also shed light on the effects of multiple recombination events on recombination detection of sequences from fast evolving pathogens. Hence, the temporal nature of the data will clearly influence the strategy used by the algorithm. While the observations made here also apply to contemporaneous data, it is the temporal distribution of the data that will best explain the prevalence or absence of recombination signals in a given sequence.

2 Methods

Algorithm *RecIdentify* reads in a recombinant network of nodes, some of which are sampled. For each sampled sequence, referred to as the query sequence, it identifies donor nodes from an earlier sampling period.

Input. A recombinant network with edge lengths representing the evolutionary history of a set of serially-sampled nucleotide sequences (sequences are not part of input), along with sampling information.
Output. Identification of recombinants, breakpoints, and donor sequences from earlier sampling times, for each sampled sequence. A *donor* sequence is the closest ancestral sequence for some part (or the entire length) of a sampled descendant sequence.

The algorithm *RecIdentify* is composed of two phases.

Phase 1. In a preliminary bottom-up traversal of the network, for each node (both sampled and unsampled), the closest sampled descendants for each sampling time are computed and stored in a list, *DescendantsInfo*.
Phase 2. Subsequently, the sequence at each sampled node is classified as being recombinant or not and its donor sequence(s) and breakpoints are identified.

2.1 Notations and Definitions

The Network. A recombinant phylogenetic network of serial samples is a directed acyclic graph/network, in which nodes are associated with nucleotide sequences and sampling times (numerical values) and edges are associated with length values. The direction of every edge is from parent node to child node. As defined below, the nodes in the network may be sampled or unsampled, with either tree nodes or recombinant nodes. When no recombination events are present, the network contains only tree nodes and is a binary tree. Figure 1 shows an example of a recombinant network.

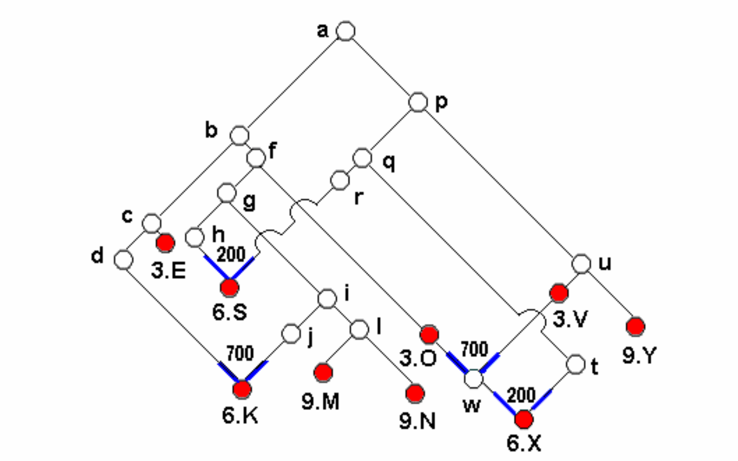


Fig. 1. Recombinant network of serially-sampled data with 4 recombination events. Sampled nodes are represented by filled circles, unsampled nodes by open circles. Each node is labeled in alphabetical order in the reverse of the order in which the node is processed during phase 1. Each sampled node is identified by a sequence name that starts with a numerical prefix indicating the sampling time point (3, 6, or 9), followed by a dot and an identifying uppercase letter. Recombinant nodes are shown with two thick edges leading into them. The numbers above them indicate the corresponding breakpoint position.

Network properties

- **A network node** may be sampled or unsampled, and is associated with a nucleotide sequence. All sequences are assumed to have the same length.
- **Sampled nodes** are assigned to a sampling time and are identified by an ID whose prefix indicates the sampling time point.
- A **tree node** has only one parent node and at most two children.
- A **recombinant node** is a node at which a recombination event takes place. It has a left and a right parent node and a corresponding breakpoint position. A recombinant node represents the evolutionary event of recombination, in which one part of the sequence (i.e., left of the breakpoint) is inherited from one parent (referred to as the left parent), and the other part (i.e., right of the breakpoint) is inherited from the other parent (referred to as the right parent).
- A **recombinant sequence** is a sequence associated with a node (either tree or recombinant node) resulting from one or more recombination events in its history.
- **A leaf node** is a node with no children nodes.

- **Edge Distance:** The distance value associated with an edge of the network represents the amount of evolutionary divergence between the two incident nodes.

A phylogenetic network is completely specified by its topology and the set of all edge lengths.

Network paths. Assume that node x is one of the parents of recombinant node y in the network. Then, depending on the location of the breakpoint, node x is a donor for only part of the sequence at node y. Thus every edge in the network from a parent to a child is associated with a portion of the sequence for which the parent is a donor. Extending this notion, we define a **path** from node v to w in the network to be a sequence of nodes (or edges) along with a specified portion of the sequence (given by a range, i.e., a start point and an end point, in the sequence). More precisely,

$$\text{path}(v,w,s) = (P, s)$$

is an ordered pair with a sequence of vertices $P = \langle n_1,..n_p\rangle$ (with the first and last entries of the sequence being v and w, respectively), and an interval range s = [start,end] indicating a start and end point of the sequence linking the nodes v and w.

The nucleotides under consideration are given by the interval range s = [start,end] often flanked by breakpoints on one or both sides. If the given recombinant network has no recombinant nodes, i.e., the given network is a tree, every path is associated with the entire sequence. In a recombinant network, there may be more than one sequence of nodes from a node v to node w, and each must be associated with a disjoint interval range, i.e., portion of the sequence.

Examples. In Figure 1, path(a,w,[0,700]) = (‹a,b,f,3.O,w›, [0,700]) and path (a,w,[701,sLen]) = (‹a,p,u,3.V,w›, [701,sLen]) are two distinct paths from a to w covering disjoint interval ranges of the sequences. Note that the breakpoint is between the 700-th nucleotide and the following one, and sLen is the length of all the sequences. Also note that paths are not directed.

Donor and Recipient lists. As mentioned before, a sampled node with recombinant nodes among its ancestors may have any number of donors. The *DonorList* can be used to store the donor information of any given node. The list contains only one network node if only one donor has been identified for the entire sequence. If different nodes have been identified as donor nodes for different parts of the sequence, then the donor list is an ordered list of donors along with the corresponding breakpoints. The *RecipientList*, to be used in Phase I of the algorithm, is structured exactly like the *DonorList*, and is used to store a list of recipients for any given node. For node x, it stores a list of all (*recipient*) nodes for which x could be a donor.

RDL notation. In order to write down the donor or recipient list, we introduce the RDL notation in this paper, written as follows:

$$[S_1|B_1|S_2|B_2|\ldots|B_{n-1}|S_n].$$

If it denotes a donors list of node x, it expresses the recombinant history of node x as an ordered list of n sampled donor sequences and n-1 breakpoints. S_1 and S_n are the leftmost and rightmost sampled donor nodes, while S_i is the ith sampled donor node. We introduce the concept of a NULL node, which is represented by a dash "-". It is used to indicate that no donor has (yet) been found for some portion of a sequence.

Examples. During phase 2 (described later) when searching for the donor list of node 9.M in Figure 1, the algorithm traverses upward to the root. When it is at node j, based on the information collected from the subnetwork reachable from node i, the donor is: [—|700|6.K], implying that no donor has yet been found for the first 700 nucleotides. However, when node b is reached, based on the information collected from the subnetwork reachable from node b, the donor list of node 9.M contains the complete information: [3.E|700|6.K].

2.2 Selection Criteria

Ancestor Criterion. A key requirement that our method imposes is that for any given sequence, all potential donor sequences must have been sampled at an earlier sampling time point.

Distance Criteria. A node v is said to be closer to node u than to node w for a segment of the sequence s = [start,end], if distance(v,u,s,c) $<_c$ distance(v,w,s,c), where distance(v,u,s,c) is the length of the shortest path between nodes v and u calculated for sequence segment s using distance measure c. In order to compute distances between the sampled data, three different distance measures may be used:

- **Path length measure:** a distance measure based on path lengths,
- **Topology distance measure:** a distance measure based on number of edges on the path, and
- **Combined distance measure:** a measure that combines the path length and topology measures.

The three measures were chosen by taking into consideration the different approaches used in existing recombination detection methods [10, 13, 14].

A sequence at node v is identified as a donor sequence of the query sequence at node q if distance(v,q,s,c) = $\min_{w \in W, v \neq w}$ {distance(w,q,s,c)}, where W is the set of all sampled nodes from sampling times prior to t_q, the sampling time of q.

Path length measure. Let *Len*(*e*) be the length value associated with edge *e*. The distance between nodes v and q under the path length criterion is given by

$$\text{distance(v,q,s,lengths)} = \sum_{e_i \in path(v,q,s)} Len(e_i) .$$

This is thus the path length measure used in traditional graph theory literature. As mentioned before, the directions of the edges on the path are ignored.

Topology distance measure. The distance between sequences at nodes v and q under this measure is given by distance(v,q,s,nodes), the minimum number of edges on any path between nodes q and v.

Combined distance measure. This is simply a weighted combination of the distances according to the edge distance measure and the topology distance measure. More formally, the combined distance is given by: distance(v,q,s,combined) = distance(v,q,s,nodes)*nodePenalty + distance(v,q,s, lengths). Here nodePenalty is a penalty imposed on the number of edges on the path.

Donor Identification. In each of the above definitions of the distance measures, the donor sequence for the query sequence at node q is defined as the sequence associated with the node v that has the smallest distance(v,q,s,c) value among all nodes v, where c is one of the three distance measure criteria. In each case, ties are broken by using an alternative distance criterion. If ties persist, a candidate donor is selected arbitrarily.

Examples. Each of the three distance measures may identify different donor sequences. In Figure 1, using the topology distance measure, sequence 6.K will be identified as a recombinant with two donors for two different portions of the sequence; in RDL notation: [3.E|700|3.O]. Note that the node 9.M is from a later time period and cannot be considered as a donor node. The edge distance measure, however, will only identify one donor sequence, [3.E], as it has a smaller edge distance from the right-hand donor (node j) than the sequence at node 3.O. Depending on the nodePenalty value, the combined distance measure could identify the same donor as that with the other two distance measures, or an entirely different donor.

2.3 The DescendantsInfo List

For the following discussion, let c be one of the three distance measures defined above. With respect to a specific node x and a sequence segment s, we say that v $<_c$ w if x is *closer* to v than to w according to a distance measure c, i.e., distance(x,v,s,c) < distance(x,w,s,c). We also define the relation $\leq_s$ as follows. We say that v $\leq_s$ w if v was sampled no later than w. Finally, we say that node v *dominates* node w, if v $<_c$ w and v $\leq_s$ w. In other words, v dominates w with respect to node x, if v is closer to x and was sampled no later than w.

The *DescendantsInfo* list is an ordered list associated with each node, with one entry for each sampling time. It is sorted by sampling times, and for each sampling time, it contains, for each segment of the sequence, the closest sampled descendants of the node. More precisely, for node x, each element in its associated DescendantsInfo is a RecipientList structure and contains the closest sampled descendant node for each segment (i.e., between two successive breakpoints) of the sequence of x. Besides the identity of the closest node, it also stores the distance to the closest node using the path length measure and the topology distance measure. (Note that the combined distance measure is not stored since it can be computed from the other two measures.)

Example. In Figure 1, the *DescendantsInfo* list for node g is [6.S|200|-|700|6.K] for sampling period 6 and [9.M] for sampling period 9.

The elements of the *DescendantsInfo* lists will be used to compute the donor lists containing the recombination history of any query sequence in Phase 2. Thus, if unsampled node x has two children v and w, then the DescendantsInfo list associated with node v contains potential donors of query nodes in the subnetwork rooted at w, and vice versa.

The Dominance rule. For tree networks, we know that the RecipientsList has only one node. The DescendantsInfo structure can be further simplified using a simple rule, i.e., the dominance rule for two sampled nodes v and w with respect to an ancestor node x and a sequence segment s, which is stated as follows:

```
if ($v <_c w$) then
        if ($v \leq_s w$) then discard w,
        else retain w, but after v.
```

In other words, the dominance rule discards all dominated nodes from the *DescendantsInfo* list of x and orders the rest by increasing distance from x, such that the *DescendantsInfo* list of every node in a tree network is sorted in the order of increasing distances and decreasing sampling times. Note that if a node x is sampled, its *DescendantsInfo* list contains only one item and that is itself.

2.4 Phase 1: Storing Candidate Donor Nodes in the *DescendantsInfo* List

In phase 1, the *DescendantsInfo* list at each node in the network is computed in a bottom-up manner using a simple Depth-First Search (DFS) traversal. The objective of phase 1 is to compute, for each node v, a list of closest sampled descendants from each sampling time. This information will be made use of in Phase 2 in the following manner. If a node v is a sibling node of some node in the path from a node x to the root, then all nodes in the *DescendantsInfo* list of v that were sampled prior to x, are potential donors of x. The *DescendantsInfo* list of a node v is computed by a process of "merging" the *DescendantsInfo* lists of its children as described below in detail.

Merging procedure. During the DFS traversal, the *DescendantsInfo* list of an unsampled leaf node is initialized to NULL and that of a sampled leaf node is initialized to contain only one RecipientList, which in turn has only one entry, i.e., itself. For all other nodes, the *DescendantsInfo* list is computed by merging the *DescendantsInfo* of the left and right children of the node. Before discussing the general case, we discuss two special cases of merging.

- **Case 1 (Merging at a recombinant parent node).** Assume that the current node is a recombinant parent node with only one child. Furthermore, assume without loss of generality, that the parent is a right recombinant donor with breakpoint B. Then the merging process will add a NULL node to the left of B in every RecipientList of the *DescendantsInfo* structure of that node, while the portion of the RecipientList for which the recombinant parent is the donor (i.e., to the right of breakpoint B) will be copied over.
- **Case 2 (Merging at nodes with nonrecombinant children).** If a node has two non-recombinant children, then the *DescendantsInfo* structure of that node can be considerably simplified by applying the dominance rule.
- **Case 3 (Merging recombinant RecipientLists).** The general case of the merging process merges the two *DescendantsInfo* structures (from the two children), each of which is a list of RecipientList structures (one for each sampling time). The merging process merges the RecipientList structures for the same sampling period with the new breakpoint list being the union of the breakpoint lists of the individual RecipientLists. For each resulting segment (i.e., between successive breakpoints) of the sequence, the merge process looks at the two donors, v and w, from each list and if $v <_c w$, it retains v, the closer of the two. In the end, if the closest sequence is the same to the left and right of a particular breakpoint, then the breakpoint is removed and the list is shortened.

Examples. In Figure 1, node j is a recombinant parent with only one child. Thus, Case 1 is applied resulting in the RecipientList [-|700|6.K]. Case 2 can be applied at node l. Here the 9.M $<_c$ 9.N when c is the path length criterion. 9.M $=_c$ 9.N when using the topology criterion, however, in this case, the path length measure is applied to break the tie. The general case 3 is applied at node g, where the RecipientLists denoted by [6.S|200|-] and [-|700|6.K] are merged resulting in a list denoted by [6.S|200|-|700|6.K]. Note the NULL node in the mid section of the sequence.

Special Cases. It is worth noting that in a recombinant network, not every sampled ancestor of a recombinant node is a donor, because it is very much dependent on the location of the breakpoints. The proposed algorithm does take care of this unusual scenario. For example, in Figure 1, the *DescendantsInfo* list of node 3.V is empty although it appears to be an ancestor of a sampled node 6.X.

2.5 Phase 2: Identification of Donor Nodes

During this search phase the closest donor sequences have to be determined for each sampled query sequence. The resulting DonorList structure is returned as the answer. When searching for the donors of a query sequence, the DonorList is initialized to the empty list. Then the network is traversed along all paths from the query node to the root. At every node, the DonorList is “merged” with entries from the *DescendantsInfo* of their sibling nodes (if any) if these entries are at a closer distance than the current answer stored in DonorList. Note that the merging process described in the previous section is now being reused in this phase. The search traverses a simple path if no recombination events are encountered along the way; otherwise the search process splits up. Whenever two of these paths meet, the DonorLists are merged again. If we assume that the network has a unique root, then all the paths will meet and the process will stop. The process will also stop when the distance traveled in the network exceeds the distance of the best answer stored in DonorList, or when a node along the path is a sampled node itself.

Special Cases. During Phase 2 the donor sequences for each “query” sequence is determined. A complex network structure may obscure recombination signals and lead to ambiguous and incorrect results. Recombinant sequences may be identified as

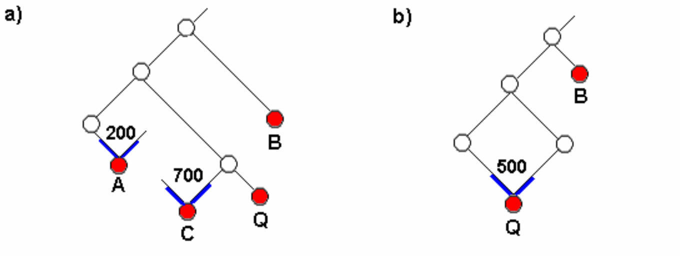


Fig. 2. Examples of ambiguous sequence identification

non-recombinants, while non-recombinants may get identified as recombinants. Figures 2(a) and (b) show examples of both these cases. In both figures, the sampled query node is marked with the label Q. In Figure 2(a), no recombination event is present in the path from Q to the root. It may nevertheless be identified as a recombinant sequence with DonorList of [A|200|B|700|C], since two closest sampled sequences from a previous sampling period (nodes A and C) are descendants of recombinant parents. In contrast, in Figure 3 (b), Q will not be identified as a recombinant sequence since the closest sampled node (node B) is not a recombinant.

2.6 Time Complexity

Phase 1. The merge process at each node merges t pairs of lists, where t is the number of sampling times. Each list can, in the worst case, have as many breakpoints r as the number of recombination events in the network. If n is the number of nodes in the network, then DFS-traversal of the network, which performs O(n) merges runs in time O(trn). Note that the number of edges in the network is linear in n (since each node has at most two edges leaving or entering). Also, note that because of recombination events, the number of nodes in the network can be arbitrarily larger than the number of leaves or number of sampled nodes in the network. In the best case scenario, n = s, all nodes are sampled nodes. Time complexity is O(n) because "every" sampled node contains itself and only itself in its DescendantsInfo list, therefore t=1 and r=1.

Phase 2. *GetClosestDonor* procedure is called for each sampled node that is not from the first sampling time point. In the worst case, this could take O(trn) for each sampled sequence and O(trsn) overall. However, this time can be improved, when during the search the result per sampling time is stored at each visited node in the path. The search will stop whenever it encounters a node that contains the result for a path that has already been searched previously. Since, with this improvement, the processing (i.e., merging) has to be performed at most once at each node, the time complexity as in Phase 1, is O(trn). Again in the best case, n=s, and the time complexity is O(n) because the donor will be found after 1 step. The donor is the sampled parent node.

3 Experimental Results and Discussion

Given a set of aligned, serially-sampled sequences and a simulated network that describes the phylogenetic relationships between the sequences, we asked the following questions in this paper: how to classify the sequences as being recombinant or non-recombinant, and how to identify donors and breakpoint positions, while using an approach that is compatible with the methodologies of published recombination detection tools. We have proposed a special algorithm, *RecIdentify*, to carry out these tasks. The resulting implementation can be conveniently used to evaluate the performance of recombination detection methods. We sought to determine the kind of recombination events that complicated the determination of the history of a sequence and how this complication affects the recombination detection process.

Testing procedure. Our findings show that the identification of recombinant sequences is greatly influenced by the choice of distance criteria, with a possible strong effect on the number of false positives detected by recombination detection programs. The program Serial NetEvolve 1.0 [12] creates a randomly generated coalescent network of serially sampled sequence data, and outputs its structure and the alignment of the sampled sequences. Sequences from both internal and terminal nodes may be present. Sequences are assigned to different sampling time points. The network output of Serial NetEvolve may be analyzed with *RecIdentify* and the results compared to the results obtained by recombination detection methods that analyze only the set of aligned sequences. A comprehensive comparison study of this kind would consist of comparing the donors and breakpoint matches in addition to computing the number of true positives and false positives and other performance measures computed using these values. While such a testing procedure is outside the scope of this current paper, it lays the groundwork for new comprehensive comparison studies.

Distance and topology approaches. For each (query) sequence the algorithm *RecIdentify* determines if the sequence has one or more donor sequences among the sampled data. Breakpoint positions are also determined for recombinant sequences. The identification process is dependent on the chosen distance criteria, which in turn is inspired by the two main classes of recombination detection methods [10, 13]. Table 1 shows the results of applying the path length (PLen), topology (Top), topology with tie-breaker (Top+TB), and combined (Com) distance criteria to 6 groups of 100 data sets. The data sets were generated with the program Serial NetEvolve [12] using 6 different recombination rates, namely 1x10^{-8}, 2x10^{-8}, and 3x10^{-8} corresponding to the low recombination rates, and 4x10^{-8}, 5x10^{-8}, and 6x10^{-8} to the high recombination rates.

Table 1. *RecIdentify* results for different distance criterias

	% recombinants		% rec donor matches						% non-rec donor matches					
Rec. rate	low	high	low			high			low			high		
Criteria			Top	Top +TB	Com	Top	Top +TB	Com	Top	Top +TB	Com	Top	Top +TB	Com
PLen	19%	39%	49%	60%	79%	42%	53%	72%	73%	79%	89%	74%	80%	89%
Top	19%	39%		75%	58%		73%	52%		92%	81%		94%	83%
Top+TB	18%	37%			75%			69%			87%			87%
Com	19%	38%												

The different distance measures (with or without tie-breaker) tend to agree in the number of recombinants found (% recombinants) for low and high recombination rates. Adding the tie-breaker to the topology measure reduces the number of sequences identified as recombinants, while adding it to the path length measure does not affect the measure in any way, as path lengths were never found to be equal. The path length and topology measures, however, return results that agree by only 49% and 42% in the choice of donor sequences for recombinant sequences (matrix of % rec donor matches) and by only 73% and 74% for non-recombinant sequences (matrix of % non-rec donor matches) as seen in Table 1.

Effects of complex phylogenetic network structures. A phylogenetic network that describes the evolutionary history of a set of recombinant taxa may obscure the actual historical evidence if too many recombination events are dispersed throughout the network. The cases discussed in the Methods section make it clear that the identification of donors in a given recombinant network is nontrivial and requires a careful algorithm that recognizes the evolutionary implications built into such a structure. Some of the more remarkable cases are summarized below:

- **Recombinant sequence may be identified as a non-recombinant.** This happens when a sequence at a recombinant node has unsampled parental nodes and the closest sampled donor is an ancestor of both unsampled parents (see Figure 2(b)).
- **Non-recombinant sequences may be identified as recombinant.** This occurs when a non-recombinant query sequence has different sequences identified as donors for different parts of the sequence, but no recombination event took place on the path from the root to the query node. (See Figure 2(a) for an example.)
- **The closest sampled nodes may not always be donor candidates of a query sequence.** If the path from the closest sampled node to the query sequence contains more than one recombination event, the query sequence may not have inherited any part of the sequence from the sampled node. (See *Special Cases* in Section 2.5)

Analysis of serially-sampled data from recombining, fast-evolving pathogens. Recombining RNA viruses are known for their immense evolutionary potential. Using *RecIdentify* to analyze networks generated by Serial NetEvolve may offer a better understanding of the evolution of such viruses, by revealing the effects that a recombinant history of a set of serially sampled viral sequences has on recombination detection and on traditional phylogenetic analysis of such data.

4 Conclusion

We present an algorithm that classifies sequences from a phylogenetic network of serial samples into two categories: recombinant or non-recombinant. For a recombinant sequence the donor sequences and breakpoint positions are also identified. The goal of the identification procedure is to use the results to achieve an evaluation of recombination detection methods that is more comprehensive than in previous studies [5-7].

The different cases that are encountered in a complex recombinant network during the identification of donors require an algorithm that recognizes the evolutionary implications built into such a structure. We show how a complex recombinant history may obscure the phylogenetic signals of the data if only a few sequences are available/sampled. Navigating the network in search of recombinant donors for a query sequence involves setting up criteria for deciding when a taxon is to be called a donor. The most influential criterion is related to the distances between sequences in the network. The distance criteria chosen in the *RecIdentify* algorithm were inspired by the two main classes of recombination detection methods, those based on sequence comparison (genetic distance) and those based on phylogenetic network topology. It is evident from the findings related to the choice of distance criteria that the selection of recombinant donors hinges greatly on the underlying distance measure used. Future work comparing recombination detection methods with respect to their choice of distance measure will reveal the magnitude of its effect on the detection process.

Acknowledgements

The work of P.B. was supported by MBRS-RISE Fellowship (NIH/NIGMS R25 GM61347). The work of G.N. was supported in part by NIH Grants P01 DA15027-01 and NIH/NIGMS S06 GM008205.

References

1. Fan, J. and D. Robertson. Links to recombinant sequence analysis/detection programs, http://bioinf.man.ac.uk/recombination/programs.shtml. 2006
2. Posada, D. and K. Crandall. The effect of recombination on the accuracy of phylogeny reconstruction. Journal of Molecular Evolution, 2002. 2002(54): p. 396–402.
3. Schierup, M. and J. Hein. Consequences of recombination on traditional phylogenetic analysis. Genetics, 2000. 156: p. 879–891.
4. Worobey, M. A novel approach to detecting and measuring recombination: new insights into evolution in viruses, bacteria, and mitochondria. Molecular Biology and Evolution, 2001. 18: p. 1425-1434.
5. Posada, D. and K.A. Crandall. Evaluation of methods for detecting recombination from DNA sequences: computer simulations. Proc. Natl. Acad. Sci. USA, 2001. 98(24): p. 13757-62.
6. Wiuf, C., T. Christensen, and J. Hein. A simulation study of the reliability of recombination detection methods. Molecular Biology and Evolution, 2001. 18(1929–1939).
7. Martin, D.P. et al. A modified bootscan algorithm for automated identification of recombinant sequences and recombination breakpoints. AIDS Research and Human Retroviruses, 2005. 21: p. 98-102.
8. Lole, K.S. et al. Full-length human immunodeficiency virus type 1 genomes from subtype C-infected seroconverters in India, with evidence of intersubtype recombination. Journal of Virology, 1999. 73(1): p. 152-60.
9. Strimmer, K. et al. A novel exploratory method for visual recombination detection. Genome Biology, 2003.
10. Martin, D.P., C. Williamson, and D. Posada. RDP2: recombination detection and analysis from sequence alignments. Bioinformatics, 2005. 21(2): p. 260-262.
11. Tsaousis, A.D. et al. Widespread recombination in published animal mtDNA sequences. Molecular Biology and Evolution, 2005. 22(4): p. 925-933.
12. Buendia, P. and G. Narasimhan. Serial NetEvolve: A flexible utility for generating serially-sampled sequences along a tree or recombinant network. Bioinformatics, 2006. 22(18): p. 2313-2314.
13. Salminen, M. et al. Identification of recombination breakpoints in HIV-1 by bootscanning. AIDS Research and Human Retroviruses, 1995. 11: p. 1423–1425.
14. Siepel, A. and B. Korber. Scanning the Database for Recombinant HIV-1 Genomes. Human Retroviruses and AIDS Compendium Part III, 1995. p. 35-60

Algorithm for Haplotype Inferring Via Galled-Tree Networks with Simple Galls
(Extended Abstract)

Arvind Gupta, Ján Maňuch, Ladislav Stacho, and Xiaohong Zhao

School of Computing Science and Department of Mathematics
Simon Fraser University, Burnaby, BC, V5A 1S6, Canada
{arvind,jmanuch,lstacho,xzhao2}@sfu.ca

Abstract. The problem of determining haplotypes from genotypes has gained considerable prominence in the research community. Here the focus is on determining sets of SNP values on individual chromosomes since such information captures the genetic causes of diseases. Present algorithmic tools for haplotyping make effective use of phylogenetic trees. Here the underlying assumption is that recombinations are not present, an assumption based on experimental results. However these results do not fully exclude recombinations and models are needed that incorporate this extra degree of complication. Recently, Gusfield studied the two cases: haplotyping via imperfect phylogenies with a single homoplasy and via galled-tree networks with one gall. In earlier work we characterized the existence of the galled-tree networks. Building on this, we present a polynomial algorithm for haplotyping via galled-tree networks with simple galls (having two mutations). In the end, we give the experimental results comparing our algorithm with PHASE on simulated data.

1 Introduction

With the great progress of the Human Genome project, recent research has focused on the problem of determining certain genome variants, SNPs, on individual chromosomes (*haplotypes*). Such information captures genetic variations and is already playing a central role in helping to determine the genetic causes of diseases and in designing effective pharmaceutical responses to these diseases ([4,18]). Experimentation allows for cost-effective determination of genotype information (the combined information of the two haplotypes for an individual across both matching chromosomes) and so the problem reduces to determining haplotypes from genotypes, the haplotyping problem. Each SNP (single nucleotide polymorphism) can take two different values over all individuals. Therefore, haplotypes can be represented with binary data, e.g., 0 representing the most common SNP value and 1 the other value. For genotypes, we use a 0 (respectively, 1) to represent that both sites are 0's (respectively, 1's), and a 2 to represent that the sites have different values (one is 0 and the other 1).

While in-vitro techniques can be used for haplotyping, the cost is prohibitive [17] and there is strong interest in the development of algorithmic tools (see

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 121–132, 2007.

[2,8,12] for survey of the problem). The first heuristic algorithm for haplotyping problem was introduced by Clark in [3]. Gusfield [7] developed the first such exact algorithms and based these on an underlying assumption of perfect phylogeny tree on SNP sequences. In particular, each SNP site mutates at most once and there are no recombinations allowed. This is based on experimental results showing that many chromosomes are blocky and nucleotides on each block tends to inherit together ([4,18]). As such these experiments do not exclude recombinations within a block and models were needed that allow for a few recombinations.

A phylogenetic network is a generalization of phylogenetic trees that allows recombinations. The problem of building a phylogenetic network with minimum number of recombinations for a set of taxa is very hard. It is intractable even for taxa with binary characters. Hein is among the first to give heuristic algorithms for building phylogenetic networks ([13,14,19]). Recently, an exact polynomial algorithm ([9]) was developed for a simplified model – the galled-tree network ([23,9]). The problem of utilizing galled-tree network model to explain evolutionary history for the original haplotypes that form genotypes was introduced by Gusfield et al. at 2003 ([10]). The problem seems very hard, and even restricted version of the problem is interesting to explore. Song et al. ([20]) designed a practical algorithm for haplotyping via galled-tree networks with one gall, as well as a polynomial algorithm for haplotyping via imperfect phylogenies with a single homoplasy. In earlier work ([6]) we considered the problem of characterizing the existence of galled-tree networks. In this paper we build on this, and present a polynomial time algorithm (with complexity $O(n^2 + nm^2)$) for haplotyping via galled-tree networks with galls having exactly two mutations. We require the genotype matrices to satisfy a natural condition which is implied, for example, by presence of at least one 1 in every column. The examination of real biological genotype data and some simulation data suggests that this condition is usually satisfied. Note that the problem of inferring haplotypes using galled-tree network generalizes the haplotyping problem via perfect phylogeny. Also note that in general the problem is very likely to be intractable [5].

Due to space limitation the proofs of correctness of Phases 4.1 and 4.2, Lemmas 2, 3 and 4 will appear in the full version of this paper.

1.1 Definitions of Phylogenetic and Galled-Tree Networks

In this paper, we consider only trees and networks with all-zero roots.

A *phylogenetic network* is a generalization of phylogenetic trees. A phylogenetic tree is built by starting with a root and repetitively adding new vertices and connecting them by mutation edges to existing vertices. Along each mutation edge the state of specified character is changed. It is assumed that each character changes at most once (the "infinite sites assumption"). To build a phylogenetic network, another operation is also allowed: add a new vertex and connect it to two existing vertices by recombination edges. The label of this vertex is a combination of labels of its two parents, prefix of one parent concatenated with the suffix of the other parent. Each recombination vertex defines a recombination cycle by tracing the first common ancestor of its parents. If a recombination cycle does not share edges with any other recombination cycle, it is called a *gall*. A network which has only galls is called a *galled-tree network*. For the formal definitions and examples see [6,11].

Given an $n \times m$ binary matrix A, we say that a phylogenetic network N with m characters *explains* A if each row of A is a label of some vertex in N.

Let S be a subset of characters. The matrix $A[S]$ is the sub-matrix of A restricted to the columns in S. By $M[S] - x$, we denote the sub-matrix of $M[S]$ from which we remove all rows whose strings are identical to x.

We say that the characters c_1 and c_2 *conflict* in a haplotype matrix M if $M[c_1, c_2]$ contains all three pairs $[0,1]$, $[1,0]$ and $[1,1]$. Note that in perfect phylogeny no two characters can conflict (four-gamete test). We assume the character set is linearly ordered, say $\{1, 2, \ldots, m\}$. The *conflict graph* G_M has the vertex set correspond to the character set and for every two characters c_1 and c_2, (c_1, c_2) is an (undirected) edge of G_M if they conflict. The following characterization of the existence of galled-tree network was obtained in [6].

Theorem 1. *([6]) Given a haplotype matrix M, M can be explained by a galled-tree network if and only if every nontrivial component (having at least two vertices) K of the conflict graph G_M satisfies the following conditions:*

*(1) K is bipartite with partitions L and R such that all characters in L are smaller than all characters in R (*the ordered component property*); and*
(2) there exists a row $x \neq 0^{|K|}$ such that $M[K] - x$ has no conflicting characters.

2 Inferring Haplotypes Via Galled-Tree Network

In this section we introduce both the perfect phylogeny haplotyping (PPH) problem and galled-tree network haplotyping (GTNH) problem.

Given a genotype $n \times m$ matrix A, find a $2n \times m$ haplotype matrix B with values in $\{0, 1\}$, where rows $2i - 1$ and $2i$ of B represent haplotypes for the genotype in row i of A. We say that B is *inferred* from A if and only if for every character $c \in \{1, \ldots, m\}$,

- if $A(i, c) \in \{0, 1\}$, then $B(2i - 1, c) = B(2i, c) = A(i, c)$; and
- if $A(i, c) = 2$, $B(2i - 1, c) \neq B(2i, c)$.

Let $\mathcal{X}_A = \{x_{r\{c_1,c_2\}};\ A(r, c_1) = A(r, c_2) = 2\}$ be the set of Boolean variables. For brevity, we will abuse notation and refer to variable $x_{r\{c_1,c_2\}}$ by $x_{rc_1c_2}$. The value of the variable $x_{rc_1c_2}$ determines the way how the pair of 2's in the row r and columns c_1 and c_2 is resolved. Define assignment $I_B : \mathcal{X}_A \rightarrow \{0, 1\}$ as follows. Let $I_B(x_{rc_1c_2}) = 0$ if the pair is resolved equally in B, i.e, $(2\ 2) \rightarrow \begin{pmatrix} 0\,0 \\ 1\,1 \end{pmatrix}$; and $I_B(x_{rc_1c_2}) = 1$ if the pair is resolved unequally in B, i.e, $(2\ 2) \rightarrow \begin{pmatrix} 0\,1 \\ 1\,0 \end{pmatrix}$. Note that specifying the assignment I_B is equivalent to specifying the matrix B (up to swapping rows $2i - 1$ and $2i$, for any $i \in \{1, \ldots, n\}$).

Given a genotype matrix A, we say that A can be *explained* by phylogenetic tree/galled-tree network if there exist a haplotype matrix B inferred from A such that B can be explained by a phylogenetic tree/galled-tree network.

Problem 1. (Perfect phylogeny haplotype (PPH) problem/Galled-tree network haplotype (GTNH) problem) Given a genotype matrix A, decide if A can be explained by a phylogenetic tree/galled-tree network.

Given a genotype matrix A, for every $x, y \in \{0,1\}$, we say that a pair of columns c_1, c_2 *induces* $[x, y]$ *in* A, if $A[c_1, c_2]$ contains at least one of the pairs $[x, y]$, $[2, y]$ and $[x, 2]$. We can define a conflict graph for genotype matrix A similarly as for haplotype matrices by relaxing the notion of conflict. In particular, character c_1 and c_2 conflict if they induce $[0, 1]$, $[1, 0]$ and $[1, 1]$ in A.

Using the characterization for the PPH problem by Bafna et al.[1], we have the following lemma about the relationship between the conflict graph of B and the assignment I_B.

Lemma 1. *Given a genotype matrix A, infer a matrix B. For every $x_{rc_1c_2}$, $x_{rc_1c_3}$, $x_{rc_2c_3} \in \mathcal{X}_A$, $I_B(x_{rc_1c_2}) + I_B(x_{rc_1c_3}) + I_B(x_{rc_2c_3}) = 0$. In addition, characters c_1, c_2 conflict in B if and only if at least one of the following is true:*

(C0) c_1, c_2 induce all three pairs: $[1, 1]$, $[0, 1]$ and $[1, 0]$ in A;
(C1) $I_B(x_{rc_1c_2}) \neq I_B(x_{r'c_1c_2})$ for some $x_{rc_1c_2}, x_{r'c_1c_2} \in \mathcal{X}_A$;
(C2) c_1, c_2 induce $[1, 1]$ in A, and there is $x_{rc_1c_2} \in \mathcal{X}_A$ such that $I_B(x_{rc_1c_2}) = 1$;
(C3) c_1, c_2 induce $[0, 1]$ and $[1, 0]$, and there is $x_{rc_1c_2} \in \mathcal{X}_A$ with $I_B(x_{rc_1c_2}) = 0$.

3 Special Instances of GTNH Problem

The GTNH problem seems to be very hard, hence we consider special instances of this problem. We will restrict the problem on genotype matrices which satisfy a structural condition called the **weak property**: *Given a genotype matrix A, we say that A has the* weak property *if every pair of characters containing* $[2, 2]$ *induces either both* $[0, 1]$ *and* $[1, 0]$*, or* $[1, 1]$*. For any two columns c_i, c_j, let* indicator *ind_{ij} be* 1 *if c_i, c_j induce* $[0, 1]$ *and* $[1, 0]$*, and* 0*, otherwise. Note that in the second case, they have to induce* $[1, 1]$.

Let us state two observations about the weak property. First, a genotype matrix with the weak property has the following useful feature: if B and B' are two haplotype matrices inferred from A such that $I_B(x_{rc_1c_2}) \neq I_{B'}(x_{rc_1c_2})$, for some $x_{rc_1c_2} \in \mathcal{X}_A$, then c_1 and c_2 conflict either in B or in B'. Second, note that the weak property is equivalent to the following condition: "every column containing 2 must also contain 1" (under assumption that A does not contain any rows with only one 2 which are easy to eliminate from the matrix without affecting the solutions to the problem). Hence, this property is not very restricting. Indeed, the data set presented in [4] and most of the simulation data sets which we randomly generated genotypes from Hudson's simulation program [15] satisfy the weak property.

In [5] we reduced the GTNH problem for matrices with the weak diagonal property (different from the weak property) to a covering problem of special hypergraphs. If this covering problem is polynomial, it would imply polynomiality of this instance of GTNH problem. However, we showed the covering problem is NP-complete. In this paper, we further simplify the problem by requiring that the solution is a galled-tree network with galls containing two mutation edges. We will call such networks *simple GTN* and the problem of deciding whether a genotype matrix can be explained by a simple GTN, the SGTNH problem. We will show that the SGTNH problem is polynomial for matrices with the weak property.

Theorem 1 gives a simple characterization of haplotype matrices which can be explained by a simple galled-tree network. Base on this we can observe that *given a haplotype matrix B, B can be explained by a simple galled-tree network if and only if every non-trivial component of the conflict graph of B has two vertices (B contains only isolated edges). Furthermore, given a genotype matrix A, if A can be explained by a simple galled-tree network then the conflict graph of A contains only isolated edges.*

Hence, from now on we will assume that the conflict graph of A has only isolated edges (otherwise, the algorithm reports *fail*). Let us call such a matrix *simple*. A haplotype matrix B explainable by an SGTN inferred from a genotype matrix A will be called a *solution*.

4 The Algorithm

Our algorithm has three phases. In the first phase, it will produce a genotype matrix A' from A (by replacing some rows with 2's with rows inferred from them) such that A' can be explained by a SGTN if and only if A does, and each row of A' has either zero, three or four 2's. In the second phase, we will further replace more rows with 2's so that a new genotype matrix A'' satisfies a special condition (SC) (described later). In the last phase, the algorithm will infer B from A'' (if possible) using a reduction to a hypergraph covering problem. Finally, a solution for A'' can be easily transformed to a solution for A.

4.1 Phase 1: Eliminating Rows with Less Than Three or More Than Four 2's.

1. *Replace every row r with one 2 (two 2's) with the $2 \times m$ matrix R inferred from r (such that $I_R(x_{rc_1c_2}) = ind_{ij}$).*
2. *For every row r with 2's in columns $c_1, c_2, \ldots, c_k$, where $k \geq 5$:*
   (a) *For every 5-tuple $c_i, \ldots, c_{i+4}$, where $i = 1, \ldots, k-4$, replace row r in $A[c_i, \ldots, c_{i+4}]$ with all (16) possible inferrings of $r[c_i, \ldots, c_{i+4}]$ to obtain 16 matrices $A^i_1, \ldots, A^i_{16}$.*
   (b) *Find the matrix $A^i_{j_i}$ whose conflict graph has only isolated edges. It can be proved that at most one of our matrices has this property. If there is no such matrix, the original inferring problem has no solution: report* fail. *Otherwise, the matrix $A^i_{j_i}$ determines unique values of $I_B(x_{rc_pc_q})$ in every solution B, for every $p, q \in \{i, \ldots, i+4\}$.*

(c) If the values of $I_B(x_{rc_pc_q})$ for $p, q \in \{1, \ldots, k\}$ are determined consistently over all 5-tuples considered, replace r by two rows inferred from r according to these values. Otherwise, there is no solution: report fail.

4.2 Phase 2: Eliminating of Some Triples of 2's.

We may assume that matrix A has zero, three or four 2's in every row. In this phase, we will resolve rows with three or four 2's, if there is a unique way of doing that. The following procedure converts matrix A' to matrix A'' with zero, three or four 2's in every row such that **(SC)**: *for every triple of 2's in one row, say in columns c_1, c_2, c_3, the sum of indicators of c_1, c_2, c_3 is 1 and the conflict graph of A'' has no edges between c_1, c_2, c_3.*

Procedure

1. *For every row with four 2's, say in columns $C = \{c_1, c_2, c_3, c_4\}$, such that at least one triple from C has the sum of indicators 0 or at least one pair from C conflicts in A', replace r by two rows inferred from r which do not violate any condition. Note that it can be proved that this inferring is unique.*
2. *For every row with three 2's, say in columns $C = \{c_1, c_2, c_3\}$, such that the sum of indicators of C equals to 0 or at least one pair from C conflict in A', replace r by two rows inferred from r which do not violate any condition. Again, this inferring is unique.*

4.3 Phase 3: Reducing to a Hypergraph Covering Problem

To complete the algorithm, we will characterize conflict graphs of all haplotype matrices inferred from the genotype matrix A'' in terms of a hypergraph coverings. Then we will use this characterization to reduce the SGTNH problem to a hypergraph covering problem. We will also provide a polynomial algorithm for the hypergraph covering problem. Let us start with the definition of a genotype hypergraph.

Given an $n \times m$ genotype matrix A with the weak property, the *genotype hypergraph* H_A of A has the set of characters $\{1, \ldots, m\}$ as a vertex set, and for every row r of A containing 2's, say in columns $c_1, \ldots, c_k$ ($k = 3, 4$), there is a hyperedge $e_r = \{c_1, \ldots, c_k\}$. Furthermore, for every two columns c and c' inducing $[0, 1]$, $[1, 0]$ and $[1, 1]$ in A (conflicting in A), there is a hyperedge $\{c, c'\}$ in H_A. The hypergraph H_A does not contain any other hyperedges. A hyperedge with k vertices will be also called a k-edge. Note that H_A has $O(n)$ k-edges, $k = 3, 4$ and $O(m)$ 2-edges. We say that a graph G on the vertex set $V(H_A)$ *covers* a hypergraph H_A with hyperedges of cardinality $2, 3, 4$, if G can be obtained as follows:

- for every 2-edge $\{c_1, c_2\}$ of H_A, add the edge (c_1, c_2) to G;
- for every 3-edge $\{c_1, c_2, c_3\}$ of H_A, add exactly one of the edges (c_1, c_2), (c_2, c_3) and (c_1, c_3) to G;
- for every 4-edge $\{c_1, c_2, c_3, c_4\}$ of H_A, add exactly two disjoint edges (d_1, d_2) and (d_3, d_4) such that $\{d_1, d_2, d_3, d_4\} = \{c_1, c_2, c_3, c_4\}$ to G.

Note that any graph covering the hypergraph H_A contains all edges of the conflict graph G_A. Consider the following hypergraph covering problem.

Problem 2 (Hypergraph Covering (HC) Problem). Given a hypergraph H with $2, 3, 4$-edges, determine whether there is a graph G that covers H and all its components have at most 2 vertices.

We have the following characterization of solutions to the SGTNH problem.

Lemma 2. *Let A be a simple genotype matrix, which satisfies the weak property, the (SC) property, and has zero, three, or four 2's in each row. A haplotype matrix B is a solution to the SGTNH problem for A if and only if G_B is a solution to the HC problem for genotype hypergraph H_A.*

4.4 Solving Hypergraph Covering Problem

Now, it suffices to show that the HC problem can be solved in polynomial time. The following lemma describes the easiest case which can be solved by first identifying and covering cycles, and then greedily remaining 3-edges.

Lemma 3. *If the hypergraph H contains only 3-edges and any two of them are either disjoint or share exactly one vertex, then the HC problem for H can be solved in time $O(n^2)$.*

We say that two 3-edges are *doubly incident* if they share two vertices. A set S of 3-edges is *doubly connected* if for any pair e and e' in S, there exists a sequence of consecutively doubly incident 3-edges from S starting with e and ending with e'. A *doubly connected component* is a maximal doubly connected set. Note that the remaining 3-edges in H can be uniquely partitioned into doubly connected components. We have the following observation.

Observation 1. *Let H be a hypergraph with only 3-edges and C any doubly connected component of H. Covering of any 3-edge of C inductively forces coverings of remaining 3-edges of C which will either lead to a contradiction or unique solution for C.*

Three pairwise doubly incident 3-edges are called a *doubly incident 3-edge-cycle*, cf. Figure 1(a). Figure 1(b) shows a hypergraph with a unique solution (depicted with bold 2-edges) which plays an important role in the following lemma which characterizes solutions for doubly connected components.

Lemma 4. *Let H be a hypergraph with only 3-edges without doubly incident 3-edge-cycles such that every pair of vertices belongs to at most two 3-edges. If C contains the hypergraph depicted in Figure 1(b) then there is either no way or a unique way how to cover C. Otherwise, C is a 3-edge-chain.*

Now, we are ready to attack the HC problem in the general case.

Theorem 2. *The HC problem can be solved in $O(n(n + m))$ time.*

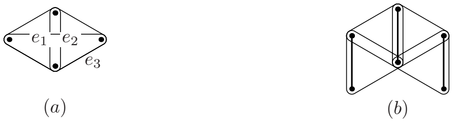


Fig. 1. (a) Doubly incident 3-edge-cycle; (b) Forcing configuration of four doubly connected 3-edges

Proof. We give a polynomial algorithm for constructing a covering graph G with components of order at most 2 from an input hypergraph H. Initially, let G be the empty graph on vertices of H. First, we remove all 2-edges from H and place them to G. The algorithm will deal with the hyperedges of H one by one using a set of rules. Each processed hyperedge is removed from H and either

(T1) some pairs of vertices contained in the hyperedge are placed as edges to G so that the conditions of Definition 4.3 are satisfied, or
(T2) the choice for pairs of vertices from this hyperedge is postponed and binded to decision on the choice for another hyperedge still in H, or
(T3) an auxiliary hyperedge is added to H and the choice for pairs from the original hyperedge is binded to decision on the choice for this new hyperedge.

The first happens only if there is a unique choice for covering the processed hyperedge. Hence, in the moment when G contains two incident edges, it follows that there is no solution to the HC problem on H. It will be obvious from the algorithm that after applying all possible rules (in given order) either all hyperedges are processed, or all unprocessed edges are 3-edges, any two unprocessed hyperedges share at most one vertex and no unprocessed hyperedge and edge of G share a vertex. Hence, we can then apply the algorithm of Lemma 3 on the modified H (containing only unprocessed edges). Union of G and a solution from the algorithm of Lemma 3 gives a solution to the HC problem.

The set of rules:

1. If there is an hyperedge e contained in another hyperedge f, we proceed as in the case (T2). From now on, we can assume that no hyperedge is contained in another.
2. If there is a pair of vertices u, v contained in at least three 3-edges, then remove all of them from H and add the edge (u, v) to G, cf. Figure 2.
3. If H contains doubly incident 3-edge-cycle, cf. Figure 1(a), remove 3-edges of the cycle from H and add a new 4-edge into H consisting of the four vertices of the cycle. It is easy to see that this does not influence the solution.

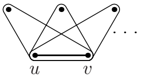


Fig. 2. The pair u, w contained in at least three 3-edges

4. For each 4-edge that intersect with other hyperedges in H or edge in G, its covering is uniquely determined or there is no covering for it and the algorithm fails. Consult Figure 3 for all possible cases. After applying the above rules as many times as possible, the only 4-edges left in H are isolated from other hyperedges and also from edges in G. Hence, for each of them, we can arbitrarily pick covering pairs of edges and remove it from H. Thus, we can assume there are no 4-edges left in H.

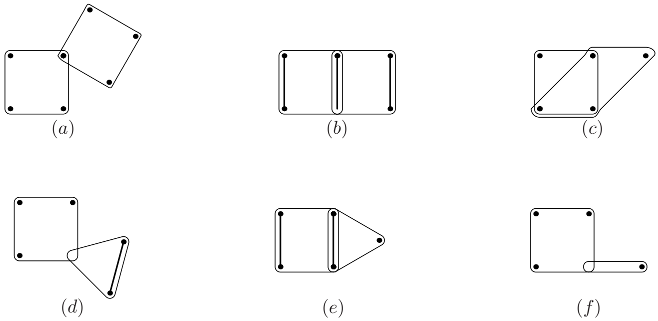


Fig. 3. An 4-edge intersecting other hyperedges. The thick edges show the covering if it exists. In cases (a), (c) and (f) there is no solution.

5. If a 3-edge intersects a 2-edge in G in a single vertex, remove the 3-edge from H and add the edge formed by the remaining two vertices of the 3-edge to G. Apply this rule whenever a new edge is added to G.
6. Consider a doubly connected component C. By Lemma 4, either there is a unique or no way of covering C, or C is a 3-edge chain. In the former case, remove C from H and add 2-edges of the solution for C to G or return fail. Assume the second case. If there is a 3-edge not in C containing an interval vertex of C, this 3-edge is uniquely covered. Hence, remove it from H and add the pair of its other two vertices as an edge to G. Now, all 3-edge chains can share only their end vertices.

If a component C is a 3-edge-chain of even length,there is a solution s for C which does not contain the end vertices, cf. Figure 4(a). Covering C in this way

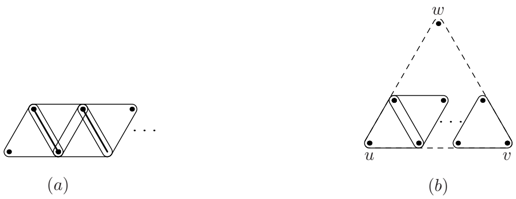


Fig. 4. (a) An example of smaller solution to a 3-edge-chain of even length. (b) Replacement of a 3-edge-chain of odd length by a new 3-edge.

will not affect the existence of the solution for H, since if there is a solution to H which contains the other solution for C then by replacing it with s results in a new solution which in addition has smaller number of edges. Hence, assume all doubly connected components are 3-edge-chains of odd length.

If there are two vertices which are end vertices of at least three 3-edge-chains, there is no solution. If there are two vertices which are end vertices of exactly two 3-edge-chains,there are only two solution for their union and each of them contains the two vertices. Hence, we can arbitrarily pick one of the solutions and add it to G and remove both 3-edge-chains from H.

Finally, we proceed as in case (T3): replace every 3-edge-chain by a 3-edge having the two end vertices u, v of the 3-edge-chain and one new vertex w, cf. Figure 4(b). The solution for the 3-edge-chain is binded to the new 3-edge as follows: if the edge (u, v) is picked to cover the 3-edge, then cover the 3-edge-chain in any of the two possible ways; if the edge (u, w) is picked, cover the 3-edge-chain by the solution that contains u; and similarly for the edge (v, w).

To analyze the running time of the algorithm, let us look at individual steps. Steps 1.–3. runs in time $O(n^2)$, 4. and 6. in $O(n)$, and 5. in time $O(nm)$. The algorithm of Lemma 3 takes time $O(n^2)$ and finally, resolution of binded hyperedges can be done in $O(n)$ time.

Note that the solution constructed by the above algorithm has the minimum number of edges, hence the corresponding galled-tree network has the minimum number of recombinations.

Complexity Analysis: The preprocessing which includes: determining values of indicators, listing of 2's in each row and computing the conflict graph of A, takes time $O(nm^2)$. Phase 1 takes time $O(nm)$, Phase 2 $O(n)$ and Phase 3 (solving HC problem) $O(n^2+nm)$. Thus, the algorithm runs in time $O(n^2+nm^2)$.

4.5 Experimental Results

We implemented the algorithm and performed the following experiments on simulation data. We compared our algorithm's performance with PHASE v2 [22,21]. The results show that our algorithm seems promising for the corresponding haplotyping problem.

In particular, the data is prepared in three steps. First, binary matrices, each having a perfect phylogeny tree, are generated using Hudson's [15] simulation program with recombination rate being 0. Then, for each matrix's perfect phylogeny tree, we randomly replace nodes on it with simple galls by adding new columns to matrices with each newly added column introducing a conflict between itself and an existing column in the matrix (thus, adding recombinations). Last, for each haplotype in the matrix, we randomly repeat it for 2 to 6 times and pair haplotypes to generate genotype matrices.

We infer haplotypes from the generated genotype matrices that have weak property using our algorithm and compare our algorithm's performance with PHASE base on three standards [22]: the *error rate*: the proportion of genotypes having more than two 2's whose haplotypes are incorrectly inferred; the

Table 1. Comparison of accuracy between our algorithm (SGTN) and PHASE on the five sets of matrices. The first row of the table lists the size of the matrices that were generated using Hudson program. The second row lists the average size of each group of matrices after applying our random algorithm and removed the duplated columns. Each rate is normalized by the number of genotype matrices in that set.

	20*20		30*20		30*40		100*50		100*100	
	30*14		45*16		75*20		199*33		199*49	
	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE	SGTN	PHASE
error rate	0.005	0.008	0.012	0.013	0.002	0.000	0.007	0.002	0.002	0.001
discrepancy	0.09	0.13	0.250	0.283	0.070	0.000	0.545	0.149	0.198	0.126
switch accuracy	0.995	0.992	0.988	0.987	1	1	0.993	0.998	0.998	0.999
time	0.54s	14.2s	0.49s	26.8s	0.43s	28.57s	0.83s	204s	2.95s	368.36s

discrepancy rate: the sum of frequency differences for each individual haplotypes (simulated or true ones); and the *switch accuracy* ([16]) of the inferred haplotypes is defined as $(h-w-1)/(h-1)$, where h is the number of 2's in a genotype g and w is the number of wrongly inferred 2's for g. Using Hudson's simulation algorithm and applying our random procedure to generate genotypes, we generated five sets of 100 matrices. The experimental results are shown in Table 1. The results show that our algorithm have comparable accuracy with PHASE while runs tens to hundreds times faster.

References

1. V. Bafna, D. Gusfield, G. Lancia, and S. Yooseph. Haplotyping as perfect phylogeny: A direct approach. *Journal of Computational Biology*, 10(3-4):323–340, 2003.
2. P. Bonizzoni, G. D. Vedova, R. Dondi, and J. Li. The haplotyping problem: An overview of computational models and solutions. *Journal of Computer Science and Technology*, 18:675–688, 2003.
3. A. Clark. Inference of haplotypes from PCR-amplified samples of dipoid populations. *Molecular Biology and Evolution*, 7:111–122, 1990.
4. M. Daly, J. Rioux, S. Schaffner, T. Hudson, and E. Lander. High-resolution haplotype structure in the human genome. *Nature Genetics*, 29(2):229–232, 2001.
5. A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. On intractability of haplotype inferring via galled-tree networks. (manuscript).
6. A. Gupta, J. Maňuch, L. Stacho, and X. Zhao. Characterization of the existence of galled-tree networks. *J. Bioinfo. and Comp. Biol.*, 4(6):1309–1328, 2006.
7. D. Gusfield. Haplotyping as perfect phylogeny: conceptual framework and efficient solutions. In *RECOMB '02: Proc. of 6th annual international conference on Computational biology*, pages 166–175. ACM Press, 2002.
8. D. Gusfield. An overview of combinatorial methods for haplotype inference. In *Computational Methods for SNPs and Haplotype Inference*, volume 2983 of *Lecture Notes in Computer Science*, pages 9–25. Springer Berlin/Verlag, 2004.
9. D. Gusfield. Optimal, efficient reconstruction of root-unknown phylogenetic networks with constrained and structured recombination. *J. Comput. Syst. Sci.*, 70(3):381–398, 2005.

10. D. Gusfield, S. Eddhu, and C. Langley. Efficient reconstruction of phylogenetic networks with constrained recombination. In *Proceedings of the 2003 IEEE CSB Bioinformatics Conference*, pages 363–374, 2003.
11. D. Gusfield, S. Eddhu, and C. Langley. Optimal, efficient reconstruction of phylogenetic networks with constrained recombination. *Journal of Bioinformatics and Computational Biology*, 2(1):173–213, 2004.
12. B. V. Halldórsson, V. Bafna, N. Edwards, R. Lipert, S. Yooseph, and S. Istrail. A survey of computational methods for determining haplotypes. In *Computational Methods for SNPs and Haplotype Inference*, volume 2983 of *Lecture Notes in Computer Science*, pages 26–47. Springer Berlin / Verlag, 2004.
13. J. Hein. Reconstructing evolution of sequences subject to recombination using parsimony. *Mathematical Biosciences*, 98:185–200, 1990.
14. J. Hein. A heuristic method to reconstruct the history of sequences subject to recombination. *Journal of Molecular Evolution*, 36:396–405, 1993.
15. R. R. Hudson. Generating samples under a Wright-Fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.
16. S. Lin, D. Cutler, M. Zwick, and A. Chakravarti. Haplotype inference in random population samples. *American Journal of Human Genetics*, 71:1129–1137, 2002.
17. R. D. Mitra, V. L. Butty, J. Shendure, B. R. Williams, D. E. Housman, and G. M. Church. Digital genotyping and haplotyping with polymerase colonies. In *Proceedings of the Nationlal Academy of Sciences of the United States of America*, volume 100, pages 5926–5931, 2003.
18. N. Patil, A. Berno, D. Hinds, W. Barrett, J. Doshi, C. Hacker, C. Kautzer, D. Lee, C. Marjoribanks, D. McDonough, B. Nguyen, M. Norris, J. Sheehan, N. Shen, D. Stern, R. Stokowski, D. Thomas, M. Trulson, K. Vyas, K. Frazer, S. Fodor, and D. Cox. Blocks of limited haplotype diversity revealed by high-resolution scanning of human chromosome 21. *Science*, 294(5547):1719–1723, 2001.
19. Y. Song and J. Hein. On the minimum number of recombination events in the evolutionary history of DNA sequences. *Journal of Mathematical Biology*, 48:160–186, 2003.
20. Y. S. Song, Y. Wu, and D. Gusfield. Algorithms for imperfect phylogeny haplotyping (IPPH) with a single homoplasy or recombination event. In R. Casadio and G. Myers, editors, *WABI*, volume 3692 of *Lecture Notes in Computer Science*, pages 152–164. Springer, 2005.
21. M. Stephens and P. Donnelly. A comparison of bayesian methods for haplotype reconstruction from population genotype data. *American Journal of Human Genetics*, 73:1162–9, 2003.
22. M. Stephens, N. Smith, and P. Donnelly. A new statistical method for haplotype reconstruction from population data. *American Journal of Human Genetics*, 68:978–989, 2001.
23. L. Wang, K. Zhang, and L. Zhang. Perfect phylogenetic networks with recombination. *Journal of Computational Biology*, 8(1):69–78, 2001.

Estimating Bacterial Diversity from Environmental DNA: A Maximum Likelihood Approach

Frederick Cohan[1], Danny Krizanc[2], and Yun Lu[2]

[1] Department of Biology,
Wesleyan University, Middletown, CT, 06459
fcohan@wesleyan.edu

[2] Department of Mathematics and Computer Science,
Wesleyan University, Middletown, CT, 06459
dkrizanc@wesleyan.edu, ylu@wesleyan.edu

Abstract. The ability to measure bacterial diversity is a prerequisite for the systematic study of bacterial biogeography and ecology. In this paper we describe a method of estimating diversity from an environmental sample of DNA and apply it to data taken from samples from the Sargasso Sea. Our approach combines the coverage depth method of Venter *et al.* [2] and the contig spectrum approach of Angly *et al.* [4], but uses maximum likelihood to recover the diversity rather than using hand-fit models as in [2]. We assume four species abundance distributions, then maximize the likelihood of fitting the coverage depth at different positions of the consensus sequence provided in the Sargasso Sea sample. The resulting estimates match well with those obtained using less mathematically rigorous approaches.

1 Introduction

The extent of prokaryote diversity has been hotly debated and rightly so. But measuring prokaryote diversity is not a trivial task [1]. There are two general approaches to estimate microbial diversity that have been applied in the past: nonparametric methods, which use detection probabilities to estimate diversity; and parametric methods, which use species abundance models to estimate diversity. Each approach has its particular strengths and limitations as well as different requirements for the input data [5].

Nonparametric methods use detection probabilities to estimate diversity. In contrast to parametric approaches, these approaches estimate OTU (operational taxonomic units) richness (the number of species in a community) from small sample sizes without assuming a particular OTU abundance model [14]. Such approaches consider the proportion of OTUs that have been observed before to those observed only once. The probability of detecting an OTU more than once will be higher in samples from less diverse communities. One disadvantage of nonparametric approaches is that they rely on estimates of the relative abundance of OTUs. Many studies have revealed that sampling biases can accompany

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 133–144, 2007.

genetic surveys of microbial diversity ([19], [20] and [21]). Another disadvantage is that they only provide a lower bound of OTU diversity. These methods do not account for very rare classes of OTUs, thus for bacterial communities, nonparametric estimators will tend to underestimate OTU diversity.

Parametric methods use species abundance models to estimate diversity. These models include the lognormal [7] and Poisson lognormal [8] among others [9]. The advantage of this method is that one can use the model to estimate diversity using relatively small samples of individuals from a given environment. However, there are several impediments to using parametric approaches to estimate microbial diversity. One limit is that there are no large data sets of microbial diversity data to support the use of any of the many competing abundance models. Another limit is that even if a compelling argument can be made in favor of a particular model, the models still require large data sets to evaluate the distribution parameters unless simplifying assumptions are made.

In this paper, we use a parametric method that combines the coverage depth method of Venter *et al.* [2] and the contig spectrum approach of Angly *et al.* [4], but uses maximum likelihood to recover the diversity rather than using hand-fit models as in the case of [2]. We use our method to estimate the number of bacterial species represented in a sample of DNA drawn from water from the Sargasso Sea. We assume four possible abundance distributions for our data, then try to recover the true diversity by maximizing the likelihood of fitting the coverage depth at every position of the consensus sequence of an environmental DNA sample.

1.1 Venter's Coverage Depth Method

A method based upon coverage depth was introduced by Venter *et al.* [2]. The method is applied to the results of whole metagenome shotgun sequencing of an environmental sample of DNA. For a single genome analysis, assembly coverage depth should be approximated by a Poisson distribution; for multiple genome analysis, assembly coverage depth should be approximated by a mixture of Poisson distributions. The empirical distribution of coverage depth at every position in the full set of assemblies was computed, then compared with hand-constructed mixtures of Poisson distributions. An excellent fit can be obtained; and to the extent that a limited range of mixtures give acceptable fits, this model may be used to estimate the diversity of the bacteria represented in the DNA extracted. However, there are obvious challenges to genome assembly in the environmental context. Additionally, the method is based on hand fitting the observed depth of coverage to a theoretical model of assembly progress for a sample corresponding to a mixture of organisms at different abundances, and is therefore ad hoc and likely less reliable.

1.2 PHACCS Methods

A method based upon the contig spectrum was introduced by Angly *et al.* [4] for estimating diversity in viral communities. They call it PHACCS (PHAge

Communities from Contig Spectrum). PHACCS uses a modified version of the Lander-Waterman model to predict a contig spectrum from assumed population parameters [4]. Since there are several genotypes in this modified model, an assumption about their underlying distribution within the community in terms of abundance has to be made. A number of distributions have been suggested including the power law, logarithmic, exponential, broken stick, niche preemption and lognomal distributions. However, predictions by PHACCS are dependent on the quality of the contig spectrum input. As a general rule, the higher the contig degree is, the better the estimations are, since the model fitting is done over a larger number of points. Additionally, the present implementation of the Lander-Waterman model assumes that all DNA fragments and all the genotypes have the same size. For these reasons, PHACCS estimates as well as ours should be only considered approximations.

2 A Maximum Likelihood Approach

We introduce our method, which essentially combines the coverage depth method and the contig spectrum method assuming a variety of abundance distributions. We calculate the likelihood of fitting the coverage depth at every position of the sequence, then recover the diversity of the sample by maximizing the log-likelihood. We use the following notation:

- S: Size of all reads sequenced from the sample
- N: Size of all the genomes in the sample
- T: Total number of genomes in the sample
- g: Average length of a genome in the sample
- k: Number of abundance levels
- a_i: the i-th abundance level ($i = 1, 2, 3...$)
- c_i: Average coverage given by the species with abundance a_i
- p_i: the proportion of the species with abundance a_i to all the species in the sample
- m_i: Number of different species with abundance a_i
- K_j: Coverage depth observed at position j of the assembly consensus sequence ($j = 1, 2, ..., J$)
- n_l: Number of positions on the assembly consensus sequence with coverage depth K_l ($l = 0, 1, ..., L$)
- J: Total number of positions on the assembly consensus sequence
- L: Total number of different coverage depth levels observed
- M: Total number of different genomes in the sample

Let x denote the fraction $\frac{S}{N}$. We have the following relations by definition:

$$J = \sum_{i=1}^{k} m_i * g = g * \sum_{i=1}^{k} m_i \tag{1}$$

and

$$T = \frac{N}{g}. \tag{2}$$

The relation among T, a_i and m_i can be expressed by the equation

$$T = \sum_{i=1}^{k} a_i * m_i. \tag{3}$$

The diversity M is

$$M = \sum_{i=1}^{k} m_i. \tag{4}$$

Since the choice of abundance distribution is very important in estimating the diversity, we will deal with the problem separately by assuming four models of the possible abundance distributions for our sample: discrete distribution with a fixed number of abundance levels, power law distribution, broken stick distribution and log-normal distribution.

2.1 Model A: Discrete Distribution

Assuming discrete distribution of abundance with a fixed number of abundance levels, sequencing of an environmental sample with more than one species should result in the sequence coverage depth reflecting a mixture of Poisson distributions. Let K_j be the coverage depth at the j-position of the assembly sequence and $P(K_j)$ be the probability of coverage depth K_j on the consensus sequence. Then the expected number of positions on the consensus sequence with coverage depth K_j is $L * P(K_j) = M * g * P(K_j)$. Let $P(K_j, a_i)$ be the probability of coverage depth K_j on the consensus sequence given by the species with abundance a_i. Then the expected number of positions on the consensus sequence with coverage depth K_j is given by $\sum_{i=1}^{k} m_i * g * P(K_j, a_i)$. Hence we have

$$M * g * P(K_j) = \sum_{i=1}^{k} m_i * g * P(K_j, a_i). \tag{5}$$

By the definition of p_i, we have

$$p_i = \frac{m_i}{\sum_{i=1}^{k} m_i} = \frac{m_i}{M} \tag{6}$$

and

$$\sum_{i=1}^{k} p_i = 1. \tag{7}$$

Dividing both sides of (5) by $M * g$, we have

$$P(K_j) = \sum_{i=1}^{k} (\frac{m_i}{M}) * P(K_j, a_i) = \sum_{i=1}^{k} p_i * P(K_j, a_i). \tag{8}$$

Assuming that the coverage depth K_j given by the species with abundance a_i satisfies the Poisson distribution, we have

$$P(K_j, a_i) = \frac{e^{-y} y^{K_j}}{K_j!} \tag{9}$$

where y is the average coverage given by the species with abundance a_i.

We assume that $y = c_i$ where c_i is the total size of fragments from species with abundance a_i divided by the genome length of the species. Then c_i is given by

$$c_i = \frac{(\frac{a_i * g}{N}) * S}{g} = (\frac{S}{N}) * a_i = x * a_i. \tag{10}$$

Then (8) can be rewritten as

$$P(K_j) = \sum_{i=1}^{k} p_i * (\frac{e^{-x*a_i} (x * a_i)^{K_j}}{K_j!}). \tag{11}$$

Hence the likelihood of fitting the coverage depth at every position of the consensus sequence is given by

$$P = \prod_{j=1}^{J} P(K_j). \tag{12}$$

Suppose there are n_l positions with the same coverage K_l for $0 \leq l \leq L$. Then we can express (12) as

$$P = \prod_{l=0}^{L} P(K_l)^{n_l} = \prod_{l=0}^{L} [\sum_{i=1}^{k} p_i * (\frac{e^{-x*a_i} (x * a_i)^{K_l}}{K_l!})]^{n_l} \tag{13}$$

where the a_i's and p_i's are the parameters.

Now we can maximize the likelihood P with respect to the parameters a_i and p_i where $a_i > 0$, $0 < p_i < 1$ for $i = 1, 2, ..., k - 1$ and $p_k = 1 - \sum_{i=1}^{k-1} p_i$. Hence m_i (the number of species with abundance a_i) can be solved using the equations (3) and (6)

$$m_i = \frac{(p_i * T)}{\sum_{i=1}^{k} (p_i * a_i)}. \tag{14}$$

Thus the diversity M is

$$M = \sum_{i=1}^{k} m_i = \sum_{i=1}^{k} \frac{(p_i * T)}{\sum_{i=1}^{k} (p_i * a_i)}. \tag{15}$$

Since P is extremely small, for practical reasons, we maximize the logarithm of the likelihood instead of the likelihood. The formula for $Log[P]$ is

$$Log[P] = \sum_{l=0}^{L} [n_l * Log(\sum_{i=1}^{k} p_i * \frac{e^{-x*a_i} (x * a_i)^{K_l}}{K_l!})] \tag{16}$$

where the a_i's and p_i's are the parameters.

2.2 Model B: Power Law Distribution

This model assumes the power law distribution of abundance. In this case, the diversity M itself is a parameter of the likelihood, so we obtain the diversity directly by maximizing the likelihood (or log-likelihood).

Let a_i denote the abundance level of the species i, then the formula for the power law distribution is

$$a_i = a * i^{-b} \tag{17}$$

for $1 \leq i \leq M$.

The parameter a represents the abundance of the most abundant genotype; b is a parameter related to the evenness (the relative abundance of individuals within a species) and M is the number of different genotypes in the community, which is the diversity we want to estimate.

A similar calculation to the above yields

$$Log[P] = \sum_{l=0}^{L} [n_l * Log \sum_{i=1}^{M} (\frac{1}{M} * \frac{e^{-x*a*i^{-b}}(x * a * i^{-b})^{K_j}}{K_j!})] \tag{18}$$

where the a, b and M are the parameters.

2.3 Model C: Broken Stick Distribution

This model assumes the broken stick distribution of abundance. Let a_i denote the abundance level of the species i, then the formula for the broken stick distribution is

$$a_i = \frac{T}{M} \sum_{q=i}^{M} \frac{1}{q} \tag{19}$$

for $1 \leq i \leq M$ where T is the total number of genomes in the sample.

A similar calculation to the above yields

$$Log[P] = \sum_{l=0}^{L} [n_l * Log \sum_{i=1}^{M} (\frac{1}{M} * \frac{e^{-x*\frac{T}{M}\sum_{q=i}^{M}\frac{1}{q}}(x * \frac{T}{M} \sum_{q=i}^{M} \frac{1}{q})^{K_j}}{K_j!})] \tag{20}$$

where M is the only parameter.

2.4 Model D: Log-Normal Distribution

The model assumes the Log-normal distribution of abundance. Let a_i denote the abundance level of the species i, then the formula for the lognormal distribution is

$$a_i = \frac{e^{m_i \sigma}}{\sum_{j=1}^{M} e^{m_j \sigma}} \tag{21}$$

where $m_i = \frac{M}{\sqrt{2\pi}}(e^{-{t_i}^2/2} - e^{-t_i+1^2/2})$.

Here $t_1 = -\infty$, $t_{i+1} = \sqrt{2}erf^{-1}[\frac{2}{M}+erf(\frac{t_i}{\sqrt{2}})]$ and $t_{M+1} = +\infty$ for $1 \leq i \leq M$ where erf is the error function ($erf(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}\, dt$) and erf^{-1} its inverse.

A similar calculation to the above yields

$$Log[P] = \sum_{l=0}^{L}[n_l * Log(\sum_{i=1}^{M}(\frac{1}{M} * \frac{e^{-x*\frac{e^{m_i\sigma}}{\sum_{j=1}^{M} e^{m_j\sigma}}}(x * \frac{e^{m_i\sigma}}{\sum_{j=1}^{M} e^{m_j\sigma}})^{K_j}}{K_j!}))] \tag{22}$$

where σ, M are the parameters.

3 Sargasso Sea Data

We tested our models on data obtained from Sargasso Sea water samples. The cell counts imply approximately one billion cells per liter, while the relative abundance of the most common organism ranged from 3% to 12% of the total. The total sequences from samples were pooled and assembled to provide a single master assembly. The empirical distribution of coverage depth at every position in the full set of assemblies was computed.

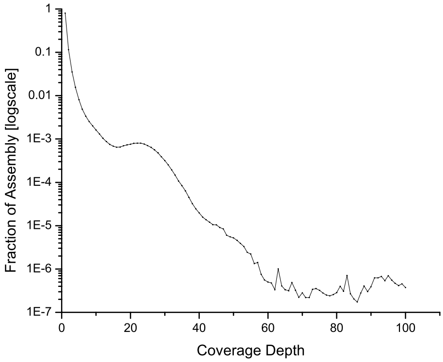


Fig. 1. Fraction of consensus sequence f_l with coverage depth K_l

The total size of the reads in Sargasso Sea water sample is

$$S = 1.66 * 10^6 * 818 = 1.36 * 10^9. \tag{23}$$

and the total number of base pairs N in the samples is

$$N = 2 * 10^{11} * (170 + 340 + 250 + 170) * 2 * 10^6/200 = 1.86 * 10^{18} \tag{24}$$

with average genome size $g = 2 * 10^6$bp/genome.

Hence the total number of genomes T is

$$T = \frac{N}{(2 * 10^6)} = 9.3 * 10^{11}. \tag{25}$$

Let x be the fraction of S to N, then

$$x = \frac{S}{N} = 7.31 * 10^{-10}. \tag{26}$$

Fig. 1 shows the fraction of the consensus sequence f_l as the y-axis and the coverage depth K_l as the x-axis. (Data provided by A. Halpern of the Venter Institute.)

4 Our Results

We assume the four species abundance distributions: models A through D, then maximize the likelihood of fitting the coverage depth at different positions of the consensus sequence provided in the Sargasso Sea Sample. **Table 1** lists the corresponding results.

Table 1. Estimates assuming different distribution model for Sargasso Sea data

Model	Abundance Levels	M	$MaxLog[P]$
Model A	one	909.4	$-1.2994 * 10^9$
	two	924.0	$-0.9984 * 10^9$
	three	997.3	$-0.9983 * 10^9$
	four	951.0	$-0.9797 * 10^9$
Model B		3504.0	$-0.9888 * 10^9$
Model C		871.0	$-1.0996 * 10^9$
Model D		917.0	$-1.0287 * 10^9$

Model A

We assume that the abundance distribution is a discrete distribution with a fixed number of abundance levels. We estimate the diversity by assuming different discrete abundance levels. If we assume two abundance levels, then by calculating the log-likelihood with $100 \leq M \leq 20,000$, we generate **Fig. 2** with the log-likelihood $Log[P]$ as the y-axis and the diversity M as the x-axis, after approximately $M = 924.0$, the value of $Log[P]$ is decreasing.

Model B

We assume that the abundance distribution is the power law distribution. In general, we don't know the abundance of the most abundant genotype, but we can still estimate the diversity by making a one parameter as well as b and M. By calculating the log-likelihood with $100 \leq M \leq 20,000$, we generate **Fig. 3** with the log-likelihood $Log[P]$ as the y-axis and the diversity M as the x-axis. After approximately $M = 3504$, the value of $Log[P]$ decreases slowly.

Model C

We assume that the abundance distribution is the broken stick distribution. By calculating the log-likelihood with $100 \leq M \leq 20,000$, we generate **Fig. 4** with

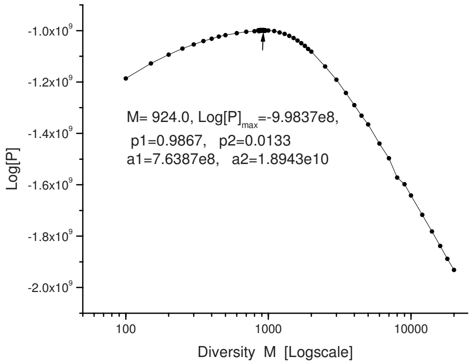


Fig. 2. $Log[P]$ at different diversity M

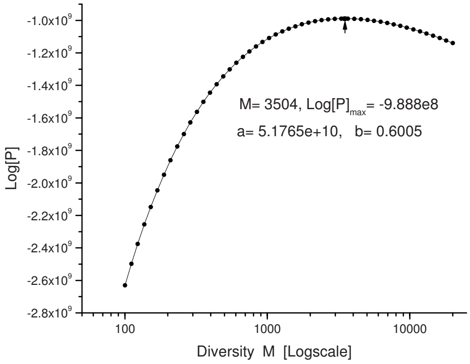


Fig. 3. $Log[P]$ at different diversity with $a = 5.0469 * 10^{10}$ and $b = 0.6001$

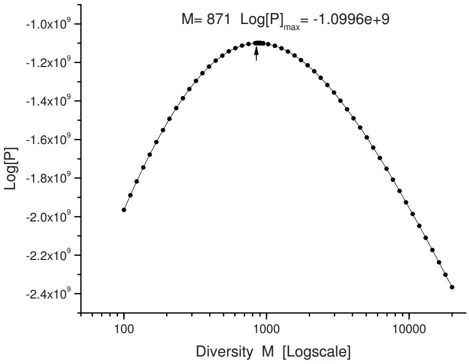


Fig. 4. $Log[P]$ at different diversity M

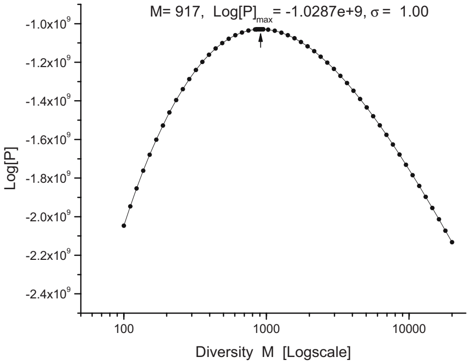


Fig. 5. $Log[P]$ at different diversity M

the log-likelihood $Log[P]$ as the y-axis and the diversity M as the x-axis. $Log[P]$ approaches its maximum when $M = 871$.

Model D

We assume the abundance distribution is the lognormal distribution. By calculating the log-likelihood assuming $\sigma = 1.0$, we generate the **Fig. 5** with the log-likelihood $Log[P]$ as the y-axis and the diversity M as the x-axis. $Log[P]$ approaches its maximum when $M = 917$.

5 Discussion

Our method is based on the method of coverage depth described in [2] and the method of contig spectrum in [4]. But unlike [2], we apply a mathematical tool of maximum likelihood estimation to maximize the likelihood of fitting the coverage depth at every position of the assembly sequence.

We assume four different abundance distributions: the discrete distribution with a fixed number of abundance levels, the power law distribution, broken stick distribution and lognormal distribution. We gave the general formulas for different cases, and developed the corresponding programs for the tests on the Sargasso Sea data. The results on the Sargasso Sea data are within the range of the estimated diversity in [2] $(1000 - 47733)$. We estimated the diversity to be approximately 900 when assuming three distributions: discrete abundance distribution, the broken stick distribution and the lognormal distribution; the diversity is estimated at approximately 3500 if the abundance distribution is the power law distribution.

We note that as in [4] our estimates are sensitive to the quality of the data provided and especially to the assembly parameters used to determine the coverage. While three out of the four distributions suggest a value close to the low end of the range estimated in [2], it has been suggested that the power law distribution is the best fit to observed diversity levels in phage data [4]. For this reason, we believe the 900 figure is best interpreted as a lower bound on the number of species present in the sample and the power law estimate is the most accurate. We plan on experimenting with more data sets in order to further evaluate this discrepancy. We also intend to work with other suggested distributions such as the logarithmic distribution, exponential distribution, and niche preemption distribution.

Acknowledgments. The authors would like to acknowledge Karin Remington and Aaron Halpern from the Venter Institute for responding to our emails and providing us with the Sargasso Sea data.

References

1. T.P. Curtis, W.T.Sloan: Exploring Microbial Diversity - A Vast Below, Science, 2005, 309: 1331–1333.
2. J.C. Venter *et al.*: Environmental Genome Shotgun Sequencing of the Sargasso Sea, Science, 2004, 304: 66–74.
3. Supporting Online Material:
URL: www.sciencemag.org/cgi/content/full/1093857/DC1.
4. F. Angly *et al.*: PHACCS, an Online Tool for Estimating the Structure and Diversity of Uncultured Viral Communities Using Metagenomic Information, BMC Bioinformatics, 2005, 6: 41. URL: www.biomedcentral.com/147-2105/6/41
5. B.J.M. Bohannan, J. Hughes: New Approaches to Analyzing Microbial Biodiversity Data, Current Opinion in Microbiology, 2003, 6: 282–287.
6. G. Myers: Whole-Genome DNA Seqencing, Computing in Science and Engineering, May-June 1999: 33–43.
7. F.W. Preston: The Commonness and Rarity of Species, Ecology, 1948, 29: 254–283.
8. M.G. Bulmer: On Fitting the Poisson Lognormal Distribution to Species Abundance Data, Biometrics, 1974, 30: 101–110.
9. S. Hubbell: The Unified Neutral Theroy of Biodiversity and Biogeography, Princeton, New Jersey: Princeton University Press; 2001
10. T.P. Curtis, W.T. Sloan, J.W. Scannell: Estimating Prokaryotic Diversity and Its Limits, Proc Natl Acad Sci USA, 2002, 99: 10494–10499.
11. J. Dunbar, S. Barns, L. Ticknor, C. Kuske: Empirical and Theoretical Bacterial Diversity in Four Arizona Soils, Appl Environ Microbiol, 2002, 68: 3035–3045.
12. J. Zhou *et al.*: Spatial and Rescource Factors Influencing High Microbial Diversity in Soil, Appl Environ Microbiol, 2002, 68: 326–334.
13. I. Kroes, P.W. Lepp, D. Relman: Bacterial Diversity Within the Human Subgingival Crevice, Proc Natl Acad Sci USA, 1999, 96: 14547–14552.
14. J.B. Hughes *et al.*: Counting the Uncountable: Statistical Approaches to Estimating Microbial Diversity, Appl Environ Microbiol, 2001, 67: 4399–4406.
15. G. Seber: The Estimation of Animal Abundance and Related Parameters, London: Griffin; 1973.
16. C. Krebs: Ecological Methodology, New York: Harper and Row; 1989.
17. A. Chao: Estimating the Population Size for Capture-recapture Data with Unequal Catchability, Biometrics, 1987, 43: 783–791.
18. M. Breitbart *et al.*: Genomic Analysis of Uncultured Marine Viral Communities, Proc Natl Acad Sci USA, 2002, 99: 14250–14255.
19. A. Reysenbach *et al.*: Differential Amplification of rRNA Genes by Polymerase Chain Reaction, Appl Environ Microbiol, 1992, 58: 3417–3418.

20. M. Suzuki, S. Giovannoni: Bias caused by Template Annealing in the Amplification of Mixutures of 16S rRNA Genes by PCR, Appl Environ Microbiol, 1996, 62: 625–630.
21. A. Speksnijder *et al.*: Microvariation Artefacts Introduced by PCR and Cloning of Closely Related 16S rRNA Gene Sequences, Appl Environ Microbiol, 2001, 67: 469–472.
22. G. Jasons, M. Wolinsky, J. Dunbar: Computational Improvements Reveal Great Bacterial Diversity and Hign Metal Toxicity in Soil, Science, 2005, 309: 1387–1390.
23. P.G. Falkowski, C. de Vargas: Shotgun Sequencing in the Sea:a Blast from the Past? Science, 2004, 304: 58–60.
24. J.M. Travis, D.R. Larsen: Meaures of Diversity, Natural Resource biometrics, 1995.

Invited Talk: Modern Homology Search

Ming Li

School of Computer Science
University of Waterloo
Waterloo, Ont. N2L 3G1
Canada
mli@cs.uwaterloo.ca

Homology search, finding similar parts between two sequences, is the most fundamental task in bioinformatics. A large fraction of the world's supercomputing time is consumed by homology search.

Traditional homology search technology is a heuristic science. Given a gene sequence, the search is either too slow (dynamic programming) or not sensitive enough. When it does return something, the results are simply some fragments of alignments.

We will talk about a new mathematical theory of optimized spaced seeds that achieves high sensitivity and high speed simulataneously for homology search, first introduced in [1], and how to achieve Smith-Waterman sensitivity with BLAST speed [2]. This methodology is now implemented in most modern homology search software.

The spaced seeds have other appliations too. We will briefly discuss how to use spaced seeds to do multiple sequence alignment [4].

We will also discuss another idea of integrating an HMM into our homology search strategy that returns structured gene matches, instead of random fragment matches [3].

This is joint work with Bin Ma, John Tromp, X.F. Cui, B. Brejova, T. Vinar, D. Shasha H. Lin, and Z.F. Zhang.

References

1. B. Ma, J. Tromp, and M. Li. PatternHunter: Faster and more sensitive homology search. *Bioinformatics*, 18:3(2002), 440-445.
2. M. Li, B. Ma, D. Kisman and J. Tromp. PatternHunter II: highly sensitive and fast homology search. *J. Bioinformatics and Computational Biology*, 2:3(2004), 417–440.
3. X.F. Cui, T. Vinar, B. Brejova, D. Shasha, and M. Li. Homology search for genes, manuscript, 2007.
4. Z.F. Zhang, H. Lin, and M. Li. MANGO: a new approach to multiple sequence alignment. Manuscript, 2007.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, p. 145, 2007.

Statistical Absolute Evaluation of Gene Ontology Terms with Gene Expression Data

Pramod K. Gupta*, Ryo Yoshida*,**, Seiya Imoto**,
Rui Yamaguchi, and Satoru Miyano

Human Genome Center, Institute of Medical Science, University of Tokyo,
4-6-1 Shirokanedai, Minato-ku, Tokyo, 108-8639, Japan
yoshidar@ims.u-tokyo.ac.jp, imoto@ims.u-tokyo.ac.jp

Abstract. We propose a new testing procedure for the automatic ontological analysis of gene expression data. The objective of the ontological analysis is to retrieve some functional annotations, e.g. Gene Ontology terms, relevant to underlying cellular mechanisms behind the gene expression profiles, and currently, a large number of tools have been developed for this purpose. The most existing tools implement the same approach that exploits rank statistics of the genes which are ordered by the strength of statistical evidences, e.g. p-values computed by testing hypotheses at the individual gene level. However, such an approach often causes the serious false discovery. Particularly, one of the most crucial drawbacks is that the rank-based approaches wrongly judge the ontology term as statistically significant although all of the genes annotated by the ontology term are irrelevant to the underlying cellular mechanisms. In this paper, we first point out some drawbacks of the rank-based approaches from the statistical point of view, and then, propose a new testing procedure in order to overcome the drawbacks. The method that we propose has the theoretical basis on the statistical meta-analysis, and the hypothesis to be tested is suitably stated for the problem of the ontological analysis. We perform Monte Carlo experiments for highlighting the disadvantages of the rank-based approach and the advantages of the proposed method. Finally, we demonstrate the applicability of the proposed method along with the ontological analysis of the gene expression data of human diabetes.

Keywords: Gene Ontology, Gene Expression Data, Statistical Meta-Analysis, Fisher's Exact Test.

1 Introduction

One of the most common interests for analysis of gene expression data is to detect genes which show the changes in their expression levels between two or more classes of cells, e.g. between cancer and normal cells. A wide variety of statistical tests, e.g. t-test, F-test, have been used for identifying genes differentially

* These authors contributed equally to this work.

** Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 146–157, 2007.

expressed between the potential cellular classes. After repeating the test across the overall genes, a strength of the statistical evidence, e.g. p-value, is assigned to each gene.

In most cases, the subsequent analysis of gene expression data is to convert the statistical evidences of differentially expressed genes into a better understanding of the underlying cellular mechanism. To this end, a large number of researchers have proposed the automatic ontological analyses of gene expression data using Gene Ontology (GO) [1]. The objective is to retrieve the GO terms which are relevant to the underlying cellular mechanisms behind gene expression data. Whereas a large number of tools have been developed for this purpose, the most existing tools implement the same approach that exploits rank information of genes which are ordered by strength of the statistical evidences.

GO::TermFinder [2] is one of the most commonly used tools to evaluate GO enrichment in a set of genes which are usually created by performing the gene selection under an acceptable level of significance. The method evaluates GO enrichment with the hypergeometric distribution or its approximation by the binomial distribution, and automatically lists the significant GO terms with the computed p-values. A large number of the existing ontological analyses implement such an approach, e.g. BinGO [3], GeneMerge [4]. One drawback of this approach is that the outcome of retrieving the significant GO terms is largely affected by the threshold rule that user selects. Commonly, constructing an appropriate threshold rule for a given multiple testing outcome is very difficult, and it is very risky to draw a conclusion solely based on the testing outcome.

Alternatively, Al-Shahrour et al. [5,6] developed a novel testing method to find the relevant GO terms based on the rank statistics of all genes which are ordered by the statistical evidences, and the web-based software referred to as FatiGO. Unlike GO::TermFinder, FatiGO does not require to draw a subset of all genes. FatiGO evaluates a GO term by repeating the test of independence in 2×2 contingency tables in which the all genes are categorized by the GO term annotations and the rank statistics. However, the application of FatiGO often leads to the serious false discovery. Particularly, the method wrongly judges the GO term as significant although all genes annotated by the GO term are unrelated to the underlying cellular mechanisms, i.e. no genes are differentially expressed. Such a drawback is closely related to the fact that rank statistics offer just a relative position of a gene among the entire gene list. This aspect will be fully discussed in this paper.

In order to overcome the drawbacks of the existing methods, we developed a new testing procedure. The proposed method has the theoretical basis on the statistical meta-analysis. It provides us a natural way of incorporating statistical evidences of the genes annotated by a GO term into a total evidence, i.e. the integrated p-value. One distinct feature of our method is to exploit strength of the statistical evidences of genes for evaluating significance of the GO term whereas the most existing methods ignore them by using only the rank statistics of genes. We point out the disadvantages of the rank-based methods and show how the proposed testing procedure overcomes them along with the Monte Carlo experiments. We also demonstrate the proposed method with the application to ontological analysis of the gene expression data of human diabetes [7].

2 Ontological Analysis of Gene Expression Data

2.1 Proposed Method

We present a statistical test to evaluate significance of GO terms with the observed gene expression data. Suppose that we test the null hypothesis $H_0^{(i)}$ for the ith gene where the total number of genes is denoted by d $(i = 1, \cdots, d)$. For example, to identify differentially expressed genes between case and control samples, one may apply t-test under the null hypothesis $H_0^{(i)} : \mu_0^{(i)} = \mu_1^{(i)}$ for $i = 1, \cdots, d$. Here $\mu_0^{(i)}$ and $\mu_1^{(i)}$ denote group means of the control and the case samples for the ith gene. Applying the testing procedure repeatedly across the d genes, one obtains the rank statistics of all d genes which are sorted by increasing order of the computed p-values.

Our objective is to retrieve the GO term annotations which are relevant to the underlying gene regulation mechanism on the basis of the statistical evidences. In sequel, the proposed method will be described along with the ontological analysis. However, we remark here that it can be applicable for retrieving more generic biological knowledge.

Let us denote a set of genes, which are classified in a GO term, by $\mathcal{F}$. The problem of evaluating the significance of $\mathcal{F}$ is stated by the statistical test with the null hypothesis H_0 and the alternative H_1 as follows:

$$\begin{aligned} H_0 &: H_0^{(i)} \text{ is true for all } i \in \mathcal{F}, \\ H_1 &: H_0^{(i)} \text{ is false for one or more } i \in \mathcal{F}. \end{aligned}$$

For instance, to understand functional gene regulations, one aims to retrieve GO terms in which more genes indicating the false $H_0^{(i)}$ are involved.

In order to evaluate this test, we present a testing procedure which exploits a technique of the statistical meta-analysis, known as the normal inversion method. Let p_i denote the p-values of the ith gene. The testing procedure for a $\mathcal{F}$ is then described as follows:

a) Transform p_i to the random deviate z_i according to

$$z_i = \Phi^{-1}(1 - p_i), \quad i \in \mathcal{F},$$

where Φ^{-1} stands for the inverse function of the standard normal cumulative distribution function.

b) Compute z-score by

$$z = \sum_{i \in \mathcal{F}} z_i \Big/ \sqrt{|\mathcal{F}|},$$

where $|\mathcal{F}|$ denotes the number of genes labelled by the GO term $\mathcal{F}$.

c) Compute an integrated p-value of the GO annotation $\mathcal{F}$, denoted by $p_{\mathcal{F}}$, by the reverse transformation of z-score as

$$p_{\mathcal{F}} = 1 - \Phi(z).$$

The basic theory of statistics indicates that each z_i is independently and identically distributed according to the standard normal distribution if and only if the individual null hypothesis $H_0^{(i)}$ is true. Moreover, under the assumption that the null hypotheses for all of the genes in $\mathcal{F}$ are true (H_0 is true), and independent to each other, the integrated z-score also follows the standard normal distribution. Based on these statistical properties, the integrated p-value captures the GO enrichment in the following way: Whereas the null p-values in $\mathcal{F}$ are uniformly distributed over $[0, 1]$, the alternative p-values are clustered around the region close to zero. This property can be formally stated if a set of tests is statistically unbiased. Hence, as the proportion of the alternative genes in $\mathcal{F}$ becomes larger, the computed z-score is shifted towards a higher value. Correspondingly, the integrated p-value becomes smaller.

2.2 Existing Methods

Recently, a large number of GO mining tools have been developed for identifying the relevant functional annotations on the basis of statistical evidences [2,3,5,6]. The most existing methods are classified into two approaches; (i) the test based on 2 × 2 contingency table (ii) the test based on the hypergeometric distribution. In below, we briefly summarize these two methods and point out their disadvantages.

GO::TermFinder. GO::TermFinder is currently one of the most commonly used GO mining tools to retrieve over-represented GO terms in a list of genes. The user is required to select n significant genes from all d genes by selecting an appropriate threshold value for the computed p-values. If m genes out of the selected n genes are annotated by a GO term $\mathcal{F}$, the method calculates the probability that m or more genes annotated by $\mathcal{F}$ are sampled in n genes by chance, under the assumption that the d genes contain $h(= |\mathcal{F}|)$ genes annotated by $\mathcal{F}$. The test follows the hypergeometric distribution (or the binomial distribution which is a large sample approximation of the hypergeometric distribution), and the p-value of $\mathcal{F}$ is computed by

$$p_{\mathcal{F}} = \sum_{t:t \geq m} \frac{\binom{h}{t}\binom{d-h}{n-t}}{\binom{d}{n}}.$$

Among others, BinGO [3], GeneMerge [4], are categorized into this approach.

We point out here some disadvantages of GO::TermFinder. Firstly, the computed p-value of the GO term, i.e. $p_{\mathcal{F}}$, is largely affected by the threshold rule that one applies. For example, suppose that we perform a conservative testing procedure which typically exchanges the number of false negatives instead of holding down the false positive outcomes. In such a case, it follows that a certain number of truly over-represented functional annotations are undiscovered. Moreover, even though the method provides us an indication about an over-represented GO term with the p-value scoring, it is very risky to draw a conclusion solely based on the p-value. Because the statistical sense of the p-value returned by GO::TermFinder is unclear, it should be regarded as a suggestion, and interpreted in the light of other biological evidence.

FatiGO. Al-Shahrour et al. [5,6] proposed a threshold-free GO mining method which exploits the rank information of all genes which are ordered according to strength of the statistical evidences. The order of genes can be determined with a variety of statistical evidences, for example, p-values, values of test statistics. The method originally assumed that the statistical evidences are collected in the analysis of differentially expressed genes.

Let $\{(1), \cdots, (d)\}$ be a list of ordered genes. The method is then summarized as follows:

1. Determine a set of J thresholds for the genes, denoted by i_k for $k = 1, \cdots, J$, where each threshold value takes a positive integer value in $\{1, \cdots, d\}$. Then, repeat the following procedure for $k = 1, \cdots, J$.
   - Divide the d genes into the two groups, $\mathcal{U}_k$ and $\mathcal{L}_k$, where the genes in $\mathcal{U}_k$ ($\mathcal{L}_k$) are selected such that ranks of any genes in $\mathcal{U}_k$ ($\mathcal{L}_k$) are equal or greater (less) than i_k.
   - Compute frequencies of the genes annotated by $\mathcal{F}$ among $\mathcal{U}_k$ and $\mathcal{L}_k$ where the frequencies are denoted by m_k and l_k, respectively.
   - Perform the Fisher exact test with the 2×2 contingency table in order to test the independence of the stratification by $(\mathcal{U}_k, \mathcal{L}_k)$ and $(\mathcal{F}, \mathcal{F}^c)$ where $\mathcal{F}^c$ denotes the complementary of $\mathcal{F}$. This process returns the p-value $p^k_{\mathcal{F}}$ for the kth partition.
2. Based on the $p^k_{\mathcal{F}}$, $k = 1, \cdots, J$, given in the above steps, the GO term $\mathcal{F}$ is judged as significant if and only if at least one $p^k_{\mathcal{F}}$ is less than a significance level.

In order to show some disadvantages of this method, we elucidate here the fact that the testing procedure implicitly performs the two-group comparison as follows:

$$\begin{aligned} &H_0 : H_0^{(i)} \text{ is true for all } i \in \mathcal{F} \text{ and } \mathcal{F}^c \\ &H_1 : \{H_0^{(i)} \text{ is false for } \mathcal{F}\} \cap \{\, H_0^{(i)} \text{ is true for } \mathcal{F}^c\}, \\ &\qquad \text{or inversely, } \{H_0^{(i)} \text{ is true for } \mathcal{F}\} \cap \{\, H_0^{(i)} \text{ is false for } \mathcal{F}^c\}. \end{aligned} \tag{1}$$

This hypothesis states whether the outcomes of the individual null hypotheses, $H_0^{(i)}$s, are identical or not under the stratification by the $\mathcal{F}$. The testing procedure is supported by the fact that if the null hypothesis H_0 is true, then the probability that each gene takes any particular rank is equal to $1/d$ for all genes. Accordingly, the genes are uniformly distributed over the 2×2 contingency table regardless of any partitions of the gene list. In this way, if one or more $p_{\mathcal{F}}^k$, $k = 1, \cdots, J$, take extremely small values, it can be concluded that the GO term annotation is over- or under-represented in the gene list.

Although the method allows us to avoid ambiguity in threshold process of genes, it still has the crucial disadvantage that can not distinguish over- and under-representation of GO annotation. Such a drawback is closely related to the fact that the hypothesis shown in (1) states two-group comparison. As an illustration, let us consider a situation where all genes in a GO term $\mathcal{F}$ have large p-values while many genes in some of the other GO terms, which are merged into $\mathcal{F}^c$, exhibit the small p-values as a consequence of the true GO enrichments. Then, the test statistic $m_k/|\mathcal{U}_k| - l_k/|\mathcal{L}_k|$ takes a large value, and consequently, the Fisher exact test returns the small p-values regardless to any partitions of the gene list. This drawback will be demonstrated more clearly along with the Monte Carlo simulations in the next section.

While the aim of the test is to evaluate the enrichment of a particular GO term $\mathcal{F}$, the method evaluates its significance by comparing with the complementary $\mathcal{F}^c$. On the other hand, the proposed testing procedure can overcome the drawbacks of the existing methods. By definition, the proposed method does not state the complementary set $\mathcal{F}^c$ in its hypothesis. Regarding this aspect, we refer it as the absolute statistical evaluation.

3 Numerical Experiments

3.1 Monte Carlo Simulation

We show the Monte Carlo experiments to illustrate the differences between the proposed method and FatiGO. The synthetic data consist of the two subtypes of samples, denoted by group 1 and group 2, and were generated as follows:

a) The sample size of each group is set to n and the total number of genes is denoted by d.
b) Among d genes, the first d_1 genes are differentially expressed between the two groups where the samples in groups 1 and 2 follow $N(-1, 1)$ and $N(1, 1)$, respectively. We denote these genes by S^+.
c) The next d_2 genes are expressed by following $N(0, 1)$ across the both groups. We denote them by S^-.
d) The remaining genes are differentially expressed between the two groups as follows: half of the genes is expressed according to $N(0, 1)$ and $N(1, 1)$, and rest of the genes follows $N(1, 1)$ and $N(0, 1)$ for group 1 and group 2, respectively.

The goal of the ontological analysis in this experiment is to retrieve the GO term S^+ which are relevant to the existing group structure, and identify irrelevance of the S^-. We first conducted the Monte Carlo experiment under the condition $(d, d_1, d_2) = (10000, 100, 20)$ and applied the two-tail t-test across the 10,000 genes. The computed t-statistics and the rank of genes in S^- and S^+ are shown by the gray and the black lines in Figure 1, respectively.

By following Al-Shahrour et al. [6], we set threshold values for FatiGO based on the t-statistic. In particular, we placed 50 threshold values at the evenly spaced 50 points which lie in a region from 1% to 99% points of the computed t-statistics. Figure 1 shows the p-values which were computed by applying the Fisher exact test across the 50 thresholds. FatiGO assigned a small p-value, 0.000475, to S^- whereas the proposed method judged S^- as insignificant (p-value = 0.618). Obviously, the result of FatiGO is inappropriate for the ontological analysis, and this numerical experiment illustrated its serious drawback. Definitely, the proposed method could identify irrelevance of the S^-.

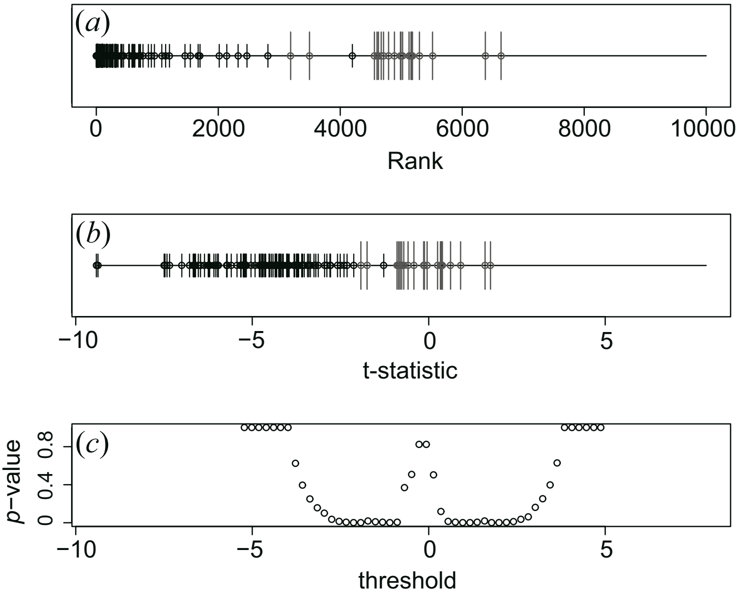


Fig. 1. Summary of the Monte Carlo experiment under $(d, d_1, d_2) = (10000, 100, 20)$: ($a$) Ranks of the genes which are placed in the order of the computed t-statistics. The genes in $\mathcal{S}^+$ and $\mathcal{S}^-$ are depicted by the black and the gray lines, respectively. (b) The t-statistics of the genes. (c) The p-values which were computed by the Fisher exact test with respect to 50 thresholds.

Next, we generated the synthetic data 100 times under the experimental parameters $(d, d_1, d_2) = (10000, 100, 10)$ and $(d, d_1, d_2) = (10000, 100, 20)$, respectively. We then implemented FatiGO and the proposed method, repeatedly. Table 1 summarizes results of these repeated experiments. While the both methods could identify the relevance of $\mathcal{S}^+$ across the every 100 experiments under the acceptance level of significance 5%, FatiGO failed to find the irrelevance of $\mathcal{S}^-$ for the most experiments. Table 1 also shows the averaged p-values of $\mathcal{S}^-$ over the 100 experiments. Regarding FatiGO, the averaged p-value of $(d, d_1, d_2) = (10000, 100, 20)$ is fairly small than that of $(d, d_1, d_2) = (10000, 100, 10)$. The results may suggest that as the number of genes in an irrelevant GO term ($\mathcal{F} = \mathcal{S}^-$) becomes larger, FatiGO tends to yield more false positive outcomes.

Table 1. The results of Monte Carlo experiments

		FatiGO	The proposed method
$d_1 = 100,\ d_2 = 10$			
S^+	≥ 0.05	0	0
	< 0.05	100	100
	mean(p-value)	4.06×10^{-60}	0#
S^-	≥ 0.05	5	97
	< 0.05	95	3
	mean(p-value)	0.0126	0.565
$d_1 = 100,\ d_2 = 20$			
S^+	≥ 0.05	0	0
	< 0.05	100	100
	mean(p-value)	3.56×10^{-64}	0#
S^-	≥ 0.05	0	93
	< 0.05	100	7
	mean(p-value)	0.00107	0.485

# This is an extremely small positive value. This results in the loss of significant digits by the numerical limitation.

3.2 Ontological Analysis of Gene Expression Data of Human Diabetes

We show here the ontological analysis of the gene expression data of human diabetic muscles [7] which were also analyzed by Al-Shahrour et al. [6] for illustrating FatiGO. The gene expression data consist of 43 samples in total, categorized into three diagnostic subgroups i.e. normal tolerance to glucose (NTG), impaired tolerance to glucose (IGT) and type 2 diabetes mellitus (DM2).

By following Al-Shahrour et al. [6], we first merged IGT and DM2 into one group, and then, apply two-tail t-test for NTG and the merged group to compute the t-statistic and p-value for each gene. We placed the 50 thresholds at the evenly spaced 50 points which lie in the region from 1% to 99% points of the computed t-statistics. We exploited Bioconductor [8] for collecting 2,592 GO terms for the molecular function and removed the genes that do not have any GO term before performing FatiGO. During this pre-screening, 17,011 genes were remained. It should be noted here that such gene screenings are not required for the proposed method.

Under the acceptable level of significance at 5%, the 188 and 712 molecular functions were identified by the proposed method and FatiGO, respectively. The both methods commonly identified the 167 molecular functions, while the 545 and 21 molecular functions were solely identified by FatiGO and the proposed method, respectively.

Table 2. List of the 40 most significant GO terms (molecular function) which were identified by the proposed method

GO term	p-value	#genes
hydrogen ion transporter activity	5.77E-08	178
monovalent inorganic cation transporter activity	6.64E-07	194
primary active transporter activity	2.92E-06	241
catalytic activity	3.78E-05	6038
carrier activity	8.08E-05	542
hydrogen-transporting ATP synthase activity, rotational mechanism	0.000107771	59
ion binding	0.000140149	4194
metal ion binding	0.000140149	4194
structural molecule activity	0.000168792	1018
cytoskeletal protein binding	0.000266417	573
cation-transporting ATPase activity	0.000270143	93
protein binding	0.000303568	6648
hydrogen-transporting ATPase activity, rotational mechanism	0.000389933	62
transferase activity	0.00044835	2103
cation binding	0.000469253	3854
lyase activity	0.000515608	206
RNA polymerase subunit kinase activity	0.000550495	2
DNA binding	0.000619234	2633
TPR domain binding	0.000627427	2
FATZ 1 binding	0.000774179	4
FATZ binding	0.000774179	4
ZASP binding	0.000774179	4
apoptogenic cytochrome c release channel activity	0.000800809	1
guanylate cyclase inhibitor activity	0.000871592	1
actin binding	0.001088743	401
ion transporter activity	0.001161264	851
cation transporter activity	0.001445382	690
NADH dehydrogenase activity	0.001452807	42
transition metal ion binding	0.001571201	2689
NADH dehydrogenase (quinone) activity	0.001726004	41
NADH dehydrogenase (ubiquinone) activity	0.001726004	41
voltage-gated ion-selective channel activity	0.001893134	6
farnesyl-diphosphate farnesyltransferase activity	0.002107711	2
5S rRNA binding	0.002224114	4
glucose-6-phosphatase activity	0.002399207	2
structural constituent of ribosome	0.002400416	257
acetylcholine binding	0.002461352	29
voltage-gated anion channel porin activity	0.002548535	5
oxidoreductase activity, acting on NADH or NADPH	0.002677663	48
zinc ion binding	0.002797571	2180

To estimate the potential false discoveries of FatiGO, we focused on the computed t-statistics of genes annotated by the 712 significant GO terms of FatiGO. These 712 GO terms contained the statistically-suspected 75 molecular functions (approximately 10% of 712) which were selected such that the t-statistics of the genes in each GO term lie in the region from -1.5 to $+1.5$. This estimate suggests that a large proportion of the GO terms identified by FatiGO is possibly a consequence of the false positive outcomes. Obviously, the proposed method is irrelevant to such misidentification.

The propose method indicated the relevance of some interesting GO annotations to different pathways of diabetes processes. The top 40 of the significant GO terms are listed in Table 2 with the computed p-values. Current biology has reported the links between diabetes with some of the identified functional annotations, e.g. lyase activity [9,10], farnesyl-diphosphate farnesyltransferase activity [11], acetylcholine binding [12,13,14], glucose-6-phosphatase activity [15], hydrogen-transporting ATPase activity [16], cation-transporting ATPase [17]. For example, Rojas et al. [16] focused on hydrogen ATPase activity in diabetes cells and hypothesized that hydrogen ATPase is suppressed in microvascular endothelial cells from diabetic rats. By using vitro angiogenesis assays, they concluded that the hydrogen ion flux were slower in cells from diabetic than normal mode, and were suppressed by hydrogen transporting ATPase inhibitors. Besides, Gautier-Stein et al. [15] studied transcriptional regulation of the glucose-6-phosphatase genes in liver. Glucose-6-phosphatase (Glc6Pase) is the key enzyme of gluconeogenesis in the liver. They suggested that glucose production by the liver contributes to hyperglycemia in type 2 diabetes. We observed from the t-tests that the genes annotated by this GO terms were highly expressed in IGT (type 1) and DM2 (type 2) than the those in the control samples, significantly.

4 Concluding Remarks

We proposed a new testing method for the ontological analysis of gene expression data. The method was developed within the framework of the statistical meta-analysis that provides a natural way of combining the p-values computed by testing individual genes. One of the key points in this paper is to elucidate the serious drawback of the rank-based testing methods which have been commonly used for the ontological analysis. We demonstrated the crucial drawbacks of these methods not only by elaborating their statistical theory, but also by some numerical examples. We also showed that the proposed method successfully overcomes the drawback and provides appropriate evaluations for the GO terms.

Throughout this paper, the proposed testing procedure was described on a situation where one aims to retrieve functional annotations relevant to the differentially expressed genes between different phenotypes. However, it should be remarked that the method can be applicable for more generic problems, because the testing theory is independent to the type of the test for individual genes. For example, the individual test of genes can be replaced by the ANOVA, which is conventionally used in multi-class testing of gene expressions, without loss of generality. This general versatility of the proposed method indicates the direction to more advanced statistical analysis of the functional annotations, e.g. pathway-level analysis with time course gene expression data and knock-out gene expression profiles.

References

1. Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., Harris, M.A., Hill, D.P., Issel-Traver, L., Kasarskis, A., Lewis, S., Matese, J.C., Richardson, J.E., Ringwald, M., Rubin, G.M., Sherlock, G.: Gene Ontology: Tool for the Unification of Biology. Nat. Genet. **25** (2000) 25–29
2. Boyle, E.I., Weng, S., Gollub, J., Jin, H., Botstein, D., Cherry, J.M., Sherlock, G.: GO::TermFinder–Open Source Software for Accessing Gene Ontology Information and Finding Significantly Enriched Gene Ontology Terms Associated with a List of Genes. Bioinformatics **20(18)** (2004) 3710–3715
3. Maere, S., Heymans, K., Kuiper, M.: BiNGO: A Cytoscape Plugin to Assess Overrepresentation of Gene Ontology Categories in Biological Networks. Bioinformatics **21(16)** (2005) 3448–3449
4. Castillo-Davis, C.I., Hartl, D.L.: GeneMerge: Post-Genomic Analysis, Data Mining, and Hypothesis Testing. Bioinformatics **19(7)** (2003) 891–892
5. Al-Shahrour, F., Diaz-Uriarte, R., Dopazo, J.: FatiGO: A Web Tool for Finding Significant Associations of Gene Ontology Terms with Groups of Genes. Bioinformatics **20(4)** (2004) 578–580
6. Al-Shahrour, F., Diaz-Uriarte, R., Dopazo, J.: Discovering Molecular Functions Significantly Related to Phenotypes by Combining Gene Expression Data and Biological Information. Bioinformatics **21(13)** (2005) 2988–2993
7. Mootha, V.K., Lindgren, C.M., Eriksson, K.F., Subramanian, A., Sihag, S., Lehar, J., Puigserver, P., Carlsson, E., Ridderstrale, M., Laurila, E. et al.: PGC-1α-Responsive Genes Involved in Oxidative Phosphorylation are Coordinately Downregulated in Human Diabetes. Nat. Genet. **34** (2003) 267–273
8. Bioconductor http://www.bioconductor.org
9. Volvenkin, S.V., Popov, V.N., Eprintsev, A.T.: Subcellular Localization and Properties of Glyoxylate Cycle Enzymes in the Liver of Rats with Alloxan Diabetes. Biochemistry (Mosc) **64(9)** (1999) 994–999
10. Obrosova, I.G., Efimov, A.S., Velikii, N.N., Zimatkina, T.I., Moiseenok, A.G.: Enzyme Systems of the Substrate and Cofactor Supply of Hyperlipogenesis in Non-Insulin-Dependent Diabetes. Biull. Eksp. Biol. Med. **105(5)** (1988) 549–552
11. Peltola, P., Pihlajamaki, J., Koutnikova, H., Ruotsalainen, E., Salmenniemi, U., Vauhkonen, I., Kainulainen, S., Gylling, H., Miettinen, T.A., Auwerx, J., Laakso, M.: Visceral Obesity is Associated with High Levels of Serum Squalene. Obesity (Silver Spring) **14(7)** (2006) 1155–1163
12. Minami, A., Ishimura, N., Harada, N., Sakamoto, S., Niwa, Y., Nakaya, Y.: Exercise Training Improves Acetylcholine-Induced Endothelium-Dependent Hyperpolarization in Type 2 Diabetic Rats, Otsuka Long-Evans Tokushima Fatty Rats. Atherosclerosis **162(1)** (2002) 85–92
13. Yu, P.K., Yu, D.Y., Cringle, S.J., Su, E.N.: Tetrahydrobiopterin Reverses the Impairment of Acetylcholine-Induced Vasodilatation in Diabetic Ocular Microvasculature. J. Ocul. Pharmacol. Ther. **17(2)** (2001) 123–129
14. Nakamura, I., Takahashi, C., Miyagawa, I.: The Alterations of Norepinephrine and Acetylcholine Concentrations in Immature Rat Urinary Bladder Caused by Streptozotocin-Induced Diabetes. J. Urol. **148(2 Pt 1)** (1992) 423–426
15. Gautier-Stein, A., Zitoun, C., Lalli, E., Mithieux, G., Rajas, F.: Transcriptional Regulation of the Glucose-6-Phosphatase Gene by cAMP/vasoactive Intestinal Peptide in the Intestine. Role of HNF4alpha, CREM, HNF1alpha, and C/EBPalpha. J. Biol. Chem. **281(42)** (2006) 31268–31278

16. Rojas, J.D., Sennoune, S.R., Martinez, G.M., Bakunts, K., Meininger, C.J., Wu, G., Wesson, D.E., Seftor, E.A., Hendrix, M.J., Martinez-Zaguilan, R.: Plasmalemmal Vacuolar H+-ATPase is Decreased in Microvascular Endothelial Cells from a Diabetic Model. J. Cell Physiol. **201(2)** (2004) 190–200
17. Iannello, S., Milazzo, P., Belfiore, F.: Animal and Human Tissue Na,K-ATPase in Obesity and Diabetes: A New Proposed Enzyme Regulation. Am. J. Med. Sci. **333(1)** (2007) 1–9
18. Richardson, M.D., Kilts, J.D., Kwatra, M.M.: Increased Expression of Gi-Coupled Muscarinic Acetylcholine Receptor and Gi in Atrium of Elderly Diabetic Subjects. Diabetes **53(9)** (2004) 2392–2396

Discovering Relations Among GO-Annotated Clusters by Graph Kernel Methods*

Italo Zoppis[1], Daniele Merico[1], Marco Antoniotti[1], Bud Mishra[2], and Giancarlo Mauri[1]

[1] Dipartimento di Informatica, Sistemistica e Comunicazione
Universitá degli Studi di Milano Bicocca
Via Bicocca degli Arcimboldi 8, U7, I-20126 Milano, Italy
[2] Bioinformatics Group, Courant Institute of Mathematical Sciences
New York University, 715 Broadway, New York, NY, 10003, USA

Abstract. The biological interpretation of large-scale gene expression data is one of the challenges in current bioinformatics. The state-of-the-art approach is to perform clustering and then compute a functional characterization via enrichments by Gene Ontology terms [1]. To better assist the interpretation of results, it may be useful to establish connections among different clusters. This machine learning step is sometimes termed *cluster meta-analysis*, and several approaches have already been proposed; in particular, they usually rely on enrichments based on flat lists of GO terms. However, GO terms are organized in taxonomical graphs, whose structure should be taken into account when performing enrichment studies. To tackle this problem, we propose a kernel approach that can exploit such structured graphical nature. Finally, we compare our approach against a specific flat list method by analyzing the cdc15-subset of the well known Spellman's Yeast Cell Cycle dataset [2].

1 Introduction

The biological interpretation of large-scale gene expression data is one of the challenges in bioinformatics [3]. The state-of-the-art approach is to perform clustering, in order to group together genes with similar expression profiles across experiments; then, in order to provide a functional characterization [4], enrichments of Gene Ontology [1] terms are computed for each cluster. In fact, it is expected that groups of genes which perform a common function also behave in a coordinated fashion. In addition, since different processes may contribute to a common function, they could be associated to the same cluster (for instance, both DNA repair and cell cycle arrest genes are both induced after DNA damage). To further assist the result interpretation, it may be useful to establish connections between different clusters; that is especially useful if they refer to different tissues, or if they have been produced according to different experimental techniques. This machine learning task can be termed *cluster meta-analysis*,

* This work has been supported by EC "Marie Curie" grant MIRG-CT-2005-031140.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 158–169, 2007.

and several approaches have already been proposed even if they usually rely on comparing enrichments built out of flat lists of GO terms [5]. However, Gene Ontology terms are not mutually independent, but organized according to a graph of taxonomical relations, and thus a flat list comparison approach may fail to exploit this specific structured nature.

A particularly interesting problem of cluster meta-analysis arises when studying time series expression data coming from microarray experiments (as in [6,7]). In this case, clustering the entire profiles may not allow some temporally localized relationships among genes to be detected. It may happen indeed that some processes are associated (that is, their genes behave in a coordinate fashion) only within a limited sequence of time-steps. Splitting microarray gene expression time series into shorter time-windows which can be clustered in separated groups has been proposed in [8,9,10] and implemented in the GOALIE system [11]. The GOALIE system provides a number of visualizers for navigating the set of relationships induced by the GO enrichments of the clustering of the time-windows. In order to compare these time-windows clusters, which are obtained in a manner very similar to the one implemented in GOALIE, it is necessary to take into account the different graph structures of their GO enrichments (*GO window graph*), we propose to use a *kernel measure* of similarity. Because of their theoretical potential as well as wide-range of applicability, kernel methods [12,13] have proven to be among the currently most successful learning algorithms. These methods work by embedding the data into a new *features* space and then looking for relations between the data in that space. In this way complex relations can be simplified and then used, for example, for classification, regression, clustering, etc.

The paper is organized as follows: in Section 2 we give a brief overview of the Kernel Methods. In Section 3, we more formally address the underlying biological problem and apply a valid kernel function to measure the similarities among the objects of our model. In Section 4, we discuss the numerical results of our evaluation experiments and finally in Section 5 we conclude and discuss some directions for future work.

2 Kernel Functions and Graph Kernel

Kernel methods have been successful in solving different problems in machine learning. The idea behind these approaches is to map implicitly the input data (i.e. training set) into a new feature (Hilbert) space $\mathcal{F}$ in order to find there some suitable hypothesis: in this way complex relations in the input space can be simplified and more easily discovered. The feature map Φ in question is defined by a kernel function k which allows to compute the inner product in $\mathcal{F}$ using only objects of the input space, hence without carrying out the map Φ. This is sometimes referred as the *kernel trick*.

Definition 1 (Kernel function). *A kernel is a function $K : X \times X \to \mathbb{R}$ capable of representing through $\Phi : X \to \mathcal{F}$ the inner product of $\mathcal{F}$ i.e.*

$$K(x, y) =< \Phi(x), \Phi(y) > \tag{1}$$

To assure that such equivalence exists a kernel must satisfy Mercer's Theorem. Hence, under certain conditions (for instance semi-definiteness of K), by fixing a kernel one can assure the existence of a mapping Φ and a Hilbert space $\mathcal{F}$ for which (1) holds. These functions can be interpreted as similarity measures of data objects into a (generally non linearly related) feature space, therefore, given K one can always induce a (non Euclidean) distance $d : X \times X \rightarrow \mathbb{R}$ such that:

$$d^2(x, y) = K(x, x) + K(y, y) - 2K(x, y) \tag{2}$$

While working with spaces whose objects are more structured, one may choose one of the many suitable kernels that exploit the underlying structure; e.g., for the set $\mathcal{G}$ of undirected labeled graphs[1] a suitable measure for the similarity between G_1 and G_2 counts the number of matching labeled random walks. These measures were proposed by different authors (see for instance [14,15,16,17]). Given a match, being obtained by comparing the label values associated to a pair of nodes (or edges), the (kernel) similarity between two random walks is then the product of the similarity values corresponding to the nodes and edges encountered along the walk. The kernel value of two graphs is then the sum over the kernel values of all pair of walks within these two graphs:

$$k_{graph}(G_1, G_2) = \sum_{walk_1 \in G_1} \sum_{walk_2 \in G_2} k_{walk}(walk_1, walk_2) \tag{3}$$

An elegant approach to construct such a similarity measure uses the direct product graph [15]:

Definition 2 (Direct product of two labeled graphs). *Given two labeled graphs $G_1 = (V, E), G_2 = (W, F)$ the direct product is denoted by $G_1 \times G_2$. The vertex set $V_\times$ and edge set $E_\times$ of this direct product are respectively defined as:*

$$\begin{aligned} V_\times(G_1 \times G_2) &= \{(v_1, w_1) \in V \times W : label(v_1) = label(w_1)\} \qquad (4)\\ E_\times(G_1 \times G_2) &= \{((v_1, w_1), (v_2, w_2)) \in V_\times^2(G_1 \times G_2) : \\ &\qquad (v_1, v_2) \in E \\ &\qquad \wedge\ (w_1, w_2) \in F \\ &\qquad \wedge\ label(v_1, v_2) = label(w_1, w_2)\} \end{aligned}$$

The nodes and edges $G_1 \times G_2$ have the same labels as the corresponding nodes and edges in G_1 and G_2. Based on this definition the *Random Walk Kernel* is then defined as follows.

[1] Here we use the following notation: a graph $G = (V, E)$ consists of a finite set of n vertices V denoted by $\{v_1, v_2, \ldots, v_n\}$, and a set of directed (possibly, weighted) edges $E \subseteq V \times V$. A walk w on G is a sequence of indices $(w_1, w_2, \ldots, w_{t+1})$ where $(v_{w_i}, v_{w_{i+1}}) \in E$ for all $1 \leq i \leq t$. A random walk is a walk where $\mathbb{P}(w_{i+1}|w_1, \ldots, w_i) = \mathbb{P}(w_{i+1}|w_i) = A_{w_i, w_{i+1}}$, i.e., the probability at w_i of picking w_{i+1} is directly proportional to the weight of the edge $(v_{w_i}, v_{w_{i+1}})$.

Definition 3 (Random Walk Kernel)

$$k_\times(G_1, G_2) = \sum_{i,j=1}^{|V_\times|} \left[\sum_{n=0}^{\infty} \lambda_n A_\times^n \right]_{i,j} \tag{5}$$

where $A_\times$ is the adjacency matrix of the product graph:

$$[A_\times]_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E_\times \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

Therefore the sum in (5) converges for a suitable choice of $\lambda_0, \lambda_1, \lambda_2 \ldots$ [15]. In this paper, as in [18], we compute the random walk kernel only for walks up to a predetermined length.

3 Method

Groups of genes may behave in a coordinate manner, but over a period of time, such coordination may be confined within some limited interval. Therefore, the usual clustering of the entire profiles, while useful (as exploited in [6,7]) may not allow some relationships among genes to be detected. To overcome this problem, it has already been suggested to split the time-series into shorter, partially-overlapping time-windows [8,9,10]. In this section we design *GO Window Graphs*, a kind of graphs where each node represents the functional enrichment of a cluster of genes in a specific time-window, and then we provide patterns of relations across time, by assembling the adjacent cluster pairs possessing minimal dissimilarity. Apart from the time-windows breakdown, our approach is in the vein of other works, such as [19,20].

3.1 GO Graph Model

In more details, our analysis is conducted on the sets $S_i = \{\mathcal{C}_{i,u} : u = 1, \ldots, N\}$ of N gene clusters obtained at step $i \in \{1, \ldots, M\}$ by splitting each time series in M time-window intervals. For each of these clusters we compute a labeled GO graph by attributing for each node v its GO term value (accessed as $\mathsf{label}_{GO}(v)$) and its enrichment (log) p-value (accessed as $\mathsf{label}_p(v)$).

3.2 A Kernel for GO Window Graphs

The graph kernel defined in section 2 is designed for discrete attributes; in that case two labeled nodes match whenever they share the same label values (i.e. their attribute). In our case labels are almost never completely identical since they contain the (*log*) p-values of the enrichment computations. More precisely, we apply the distance (2) induced from a specific kernel function that measures (dis)similarity between the associated functional processes, in order to determine a relationship $\mathcal{L} \subseteq S_1 \times S_2 \times \ldots \times S_M$ which can be used to link similar GO Window Graphs (i.e. clusters). One such suitable function can be constructed

- by considering in (4) a match between two vertex v_1 and v_2 if $\mathsf{label}_{GO}(v_1) = \mathsf{label}_{GO}(v_2)$, and
- by modifying as in [18] the adjacency matrix (6).

We have the following:

Definition 4. *Given two graphs $G_1 = (V, E)$ and $G_2 = (W, F)$ and two walks $walk_1 = (v_1, v_2, \ldots, v_n) \in G_1$ and $walk_2 = (v_1, v_2, \ldots, v_n) \in G_2$, with $v_i \in V$, $w_i \in W$. The walk kernel is defined as*

$$k_{walk}(walk_1, walk_2) = k_{step}((v_i, v_j), (w_i, w_j)) \tag{7}$$

for each i and j.

The random kernel is still the sum over all kernel on pairs of walk as in [18] and it can be computed with following adjacency:

$$[A_\times]_{((v_i,w_i),(v_j,w_j))} = \begin{cases} k_{step}((v_i, v_j), (w_i, w_j)) & \text{if } ((v_i, v_j), (w_i, w_j)) \in E_\times \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

with $E_\times = E_\times(G_1 \times G_2)$ and $(v_i, v_j) \in E$ and $(w_i, w_j) \in F$.

Our step kernel has a simpler formulation having the goal of comparing only the (log) p-values of the original node, the destination nodes and their respective GO terms. More formally:

Definition 5 (Step kernel for GO graphs). *For $i = 1, \ldots, n-1$, the* step kernel *is defined as*

$$\begin{aligned} &k_{step}((v_i, v_j), (w_i, w_j)) \\ &= k_{nodepv}(v_i, w_i) * k_{nodeterm}(v_i, w_i) * k_{nodepv}(v_j, w_j) * k_{nodeterm}(v_j, w_j) \end{aligned} \tag{9}$$

where for k_{nodepv} we use the Brownian bridge kernel [21]

$$k_{nodepv}(x, x') = \max(0, c - |\mathsf{label}_p(x) - \mathsf{label}_p(x')|) \tag{10}$$

and for the kernel $k_{nodeterm}$ on the GO terms, a Dirac function Kernel:

$$k_{nodeterm}(x, x') = \begin{cases} 1 & \text{if } \mathsf{label}_{GO}(x) = \mathsf{label}_{GO}(x') \\ 0 & \text{otherwise.} \end{cases} \tag{11}$$

Now, by defining $U = \bigcup_i S_i$ and by following the same proof scheme of [18], we can show that

$$k(\mathcal{C}_{i,u}, \mathcal{C}_{i+1,v}) = \sum_{j=1}^{|V_\times|} \sum_{n=0}^{\infty} \lambda_n A_\times^n \tag{12}$$

is still a valid kernel on $U \times U$. Thus, we have the following lemma.

Lemma 1. *Let k be defined as in (12). Then it is a positive definite kernel function.*

Proof. The node kernel is a Brownian bridge kernel that is known to be positive definite [21]. Since pointwise multiplication preserves positive definiteness, step kernel is consequently positive definite. By fixing now $\tilde{k}^j_{walk}$, for all these pairs of walks of length j only and zero otherwise [16] we have that $\tilde{k}^j_{walk}$ is a tensor product of step kernels for walks which is zero extended to the whole set of pair of walks, hence is positive definite. Again, K_{walk} is a sum over all $\tilde{k}^j_{walk}$ and is valid as well. The modified random walk kernel follows being a convolution kernel proven to be positive definite. Hence (12) can measure the similarity among objects in U and specifically can perform the similarity for each $x \in S_i$ and $y \in S_{i+1}$.

Since each kernel induces a distance (2), here we take $d_i : U \times U \rightarrow \mathrm{I\!R}$:

$$d(\mathcal{C}_{i,u}, \mathcal{C}_{i+1,v})^2 = k(\mathcal{C}_{i,u}, \mathcal{C}_{i,u}) + k(\mathcal{C}_{i+1,v}, \mathcal{C}_{i+1,v}) - 2k(\mathcal{C}_{i,u}, \mathcal{C}_{i+1,v}) \tag{13}$$

Therefore it becomes quite natural to link the cluster $\mathcal{C}_{i,u}$ at time i with the cluster $\mathcal{C}_{i+1,v}$ at time $i+1$ on the base of the minimal distance value i.e.

$$\mathcal{C}_{i,u} \sim \mathcal{C}_{i+1,v} \text{iff } d(\mathcal{C}_{i,u}, \mathcal{C}_{i+1,v}) = \min_{\mathcal{C}_{i,m} \in S_i, \mathcal{C}_{i+1,n} \in S_{i+1}} d(\mathcal{C}_{i,m}, \mathcal{C}_{i+1,n}) \tag{14}$$

4 Numerical Results

The purpose of the following analysis is mainly to compare the results of our application against a specific flat list approach. The n-tuples in $\mathcal{L} \subseteq S_1 \times S_2 \times ... \times S_M$ are expected to contain clusters with functional homogeneity among each other and maximum separation of functional annotations across clusters of other n-tuples. Therefore we conducted numerical evaluations to assess two quality indexes: (I) maximum density with minimum diversity within a cluster and (II) maximum separation between clusters. This reflects one of the main approaches in quality validation tests for a clustering technique. In general, DNA microarray expression data-sets are grouped with the expectation that genes with similar functional features group together. In order to fulfill this expectation, we have applied the indexes for cohesiveness from [22] while performing the clustering at each "window-time interval". We briefly report all these indexes to provide a better understanding of our results.

The probability of selecting a gene associated to a functional group (identified by a certain GO term) i within a cluster r can be estimated knowing the total number of genes in r i.e. $p_{ir} = \frac{n_i}{n_r}$ and $\sum p_{ir} = 1$. One can model the functional cohesiveness within a cluster using Shannon's information theory. Higher value of cohesiveness of a cluster is measured by a high degree of certainty that the genes in a cluster belongs to a functional group. Hence, the cohesiveness of a cluster is its information content:

$$CC = -\sum_{i=1}^{k} p_{ir} \log_2(p_{ir}) \tag{15}$$

In our application the relation $\mathcal{L} \subseteq S_1 \times S_2 \times ... \times S_M$ is discovered step by step by evaluating the kernel-induced distance between pair of clusters $\mathcal{C}_{i,u}, \mathcal{C}_{i+1,v}$ for each time-window interval i. That is, discharging the interval steps we can consider this distance as a way to group together genes of the respective clusters in the pair, where these genes share the same functional processes in an ideal case. It seems, indeed quite natural to consider the cluster cohesiveness index CC ([15]) when a cluster is defined as $\mathcal{C}_{i,u} \cup \mathcal{C}_{i+1,v}$. Therefore, the total cluster cohesiveness can be defined as

$$TCC = -\sum_{r=1}^{r=m} \sum_{i=1}^{k} p_{ir} \log_2(p_{ir}) \tag{16}$$

The functional separation across different clusters can be measured by estimating the probability b_{ir} of selecting a gene of functional group i in cluster r among all genes belonging to the functional group i, i.e. $b_{ir} = \frac{n_{ir}}{N_i}$ where n_{ir} is the total number of genes of functional group i in cluster r, N_i the total number of genes in the behavioral group i and $\sum b_{ir} = 1$. The information content of a functional group i in all the clusters reflects the specificity of the functional group and thereby indicates the separation property, more formally:

$$GC = -\sum_{r=1}^{m} b_{ir} \log_2(b_{ir}), \tag{17}$$

while the total cluster separation can be defined as

$$TGC = -\sum_{i=1}^{k} \sum_{r=1}^{m} b_{ir} \log_2(b_{ir}) \tag{18}$$

For a simple flat list approach we first removed from each cluster those terms whose p-values was below a detection threshold and then applied as in ([13]) the Jaccard distance index:

$$J(X, Y) = 1 - \frac{|X \cap Y|}{|X \cup Y|} \tag{19}$$

for each X and $Y \in \bigcup_i S_i$.

4.1 Preliminary Analysis of the Yeast Cell Cycle Data-Set

The Spellman's Yeast Cell Cycle data-set [2] includes three main experiments: cdc15, alpha-factor, elutriation (where the names correspond to the three different methods employed for cell synchronization). We have analyzed only the cdc15 subset, which is 18 time-points long.

GO annotations of *S. cerevisae* genes have been downloaded from the SGD database (`http://www.yeastgenome.org`). The GO DAG has been derived from R package GO 1.14.1. Functional Enrichment p-values have been calculated according to the hypergeometric distribution approximating Fisher's exact test (a

Table 1. The comparison of the cluster cohesiveness and separation indexes for connected clusters between time windows 1 and 2, and between time windows 2 and 3. The Kernel induced distance produces better results, although it is more expensive to compute.

Jaccard	1-2	2-3	Total
TCC	253.99	333.65	587.64
TGC	2134.9	2513.4	4648.30
Kernel	1-2	2-3	Total
TCC	276.41	303.45	579.86
TGC	2298.3	2341.3	4639.6

standard in existing resources, such as [23,24,25]). GO terms annotating less than 4 genes of all genes from the experiment ("universe-set") have been excluded from the analysis. The p-value of GO terms with less than 5 genes in sample has been arbitrarily set to 1 (not relevant at all).

The cdc15 data-set was divided into 5 time-windows, with 5 time-steps each, with one overlapping time point. We have computed 15 clusters for each of the first three time-windows using a standard k-means algorithm. Then we identified the "most similar" pairs of adjacent clusters according to (i) the Kernel induced distance and (ii) Jaccard coefficient.

Then, we have merged associated pairs clusters, obtaining new clusters. For these, we have computed cluster cohesiveness and separation indexes, TCC and TGC, which have been already described elsewhere in the paper. These display a superior performance for the Kernel induced distance over the Jaccard coefficient. Table 1 shows some of these comparisons between the Jaccard coefficient and the Kernel induced distance according to TCC and TGC.

In addition, we also ran two qualitative benchmarks to test our Kernel and Jaccard performance. We have traced connections between clusters according to:

- the absolute value of the intersection between the sets of genes,
- a manual curation.

For each connection found by Jaccard and Kernel, we specified whether it had been found or not according to the other methods.

Again, the superiority of the Kernel approach is generally confirmed, although a substantial disparity occurs between time window 1 to 2 and time window 2 to 3 couplings, with the second one displaying a greater performance difference.

5 Biological Results

The division of the Spellman Yeast Cell Cycle Data into windows of 5 time-points, yields time windows roughly corresponding to two phases each:

- window 1 (1-5): G1, S (and partially G2)
- window 2 (5-9): G2, M (and partially G1)
- window 3 (9-13): G1, S

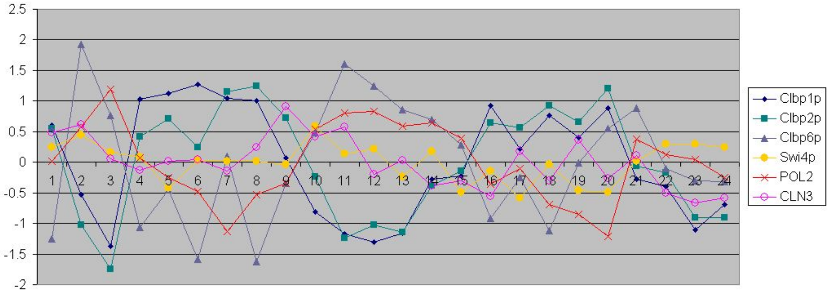


Fig. 1. Transcriptional profiles of cell cycle marker genes in [2] data, cdc15 subset. CLN3p is required for M→G1 transition, and it also indirectly activates the SBF complex. Swi4p is part of the transcription-regulating complex SBF, which binds its targets in early G1, but it is active only in late G1, and is a key player for G1→S transition (together with MBF, whose components are not reported); Clb6p is responsible for an initial inactivation (through nuclear export) of SBF and MBF during S-phase; POL2 has been assumed as a rough predictor of DNA-replication activity under S-phase; Clb1/2p are responsible for switching off SBF and MBF in G2 phase, and therefore are key players of S→G2 transition. Comparing the peaking areas of Clbp6p and POL2, and the depression areas of Clbp1/2p, it is evident that G1 and S phases are "compressed" in the initial time-steps.

Table 2. c8w1 enrichment reports these terms, ranked by their p-value

translation and ribosome subgroup	
4.70e-13	cytosolic large ribosomal subunit (sensu Eukaryota)
4.30e-11	translation
7.43e-11	structural constituent of ribosome
5.40e-07	cytosolic small ribosomal subunit (sensu Eukaryota)
1.22e-06	translational elongation
1.04e-4	ribosomal small subunit assembly and maintenance
2.60e-3	ribosome
cell wall, plasma membrane and vescicular compartments sub-group	
1.04e-04	vacuolar transport
1.77e-03	endoplasmic reticulum
2.70e-03	transporter activity
3.40e-03	integral to membrane

The discrepancy of window 1 and 2 w.r.t. holding exactly two phases is probably due to synchronization, which alters the regularity in the very initial time-points (see Figure 1 for details).

The demonstration of this statement is provided by the chart in Figure 1, showing the normalized expression levels of a few marker genes.

To provide an example of the results yielded by our method, we consider the maximal-similarity connections found among three adjacent clusters, respectively belonging to time-window 1, 2 and 3 (they will be termed c8w1, c10w2, c13w3 in Tables (2), (3), and (4)).

Therefore, a robust core of terms can be identified: (1) protein synthesis by the cytoplasmic ribosome, (2) glycolysis and glyconeogenesis, and (3) cell wall, plasma membrane and vescicular compartment.

Table 3. c10w2 enrichment reports these terms

translation and ribosome subgroup	
1.33e-07	translational elongation
9.58e-06	ribosome
1.13e-05	cytosolic large ribosomal subunit (sensu Eukaryota)
1.25e-05	cytosolic small ribosomal subunit (sensu Eukaryota)
2.61e-05	translation
4.70e-05	structural constituent of ribosome
2.97e-03	translational initiation
glycolysis and glyconeogensis subgroup	
2.62e-08	glycolysis
1.19e-06	gluconeogenesis
cell wall, plasma membrane and vescicular compartments subgroup	
3.46e-08	membrane
7.93e-05	transporter activity
3.80e-04	cell wall (sensu Fungi)
7.86e-04	transport
2.46e-03	endoplasmic reticulum
3.00e-03	plasma membrane

Table 4. c13w3 enrichment reports these terms

translation and ribosome subgroup	
1.25e-05	translational elongation
2.54e-04	ribosome
glycolysis and glyconeogensis sub-group	
4.121e-06	glycolysis
7.36e-06	gluconeogenesis
3.14e-05	hexose transport
cell wall, plasma membrane and vescicular compartments subgroup	
1.89e-05	integral to plasma membrane
2.51e-05	transporter activity
3.14e-04	cell wall (sensu Fungi)
4.71e-04	plasma membrane
6.69e-04	membrane
3.79e-03	integral to membrane

Actually, c8w1 does not include glycolysis and glyconeogenesis genes, which happen to be in a different cluster (c11w1); however, if we compare the profiles of c11w1 and c8w1, we can see that they are quite correlated, displaying a slightly increasing profile, although c11w1 is much more noisy; we probably observe this discrepancy because clustering has not been optimized employing functional annotation maximization as the objective for optimal k-means selection. We intend to include this feature in an enhanced version of our method.

The relative stability of functional annotations in these clusters suggests that regulation of basal metabolism and protein synthesis is coupled in the same fashion through all the cell cycle, without any detectable de-coupling event. Please note that these connections were not successfully found employing the alternative method based on Jaccard coefficient.

6 Conclusion

We have presented an application of *graph kernel* methods to group clusters of gene expression measurements organized over a time line. Our main contribution is an initial kernel similarity function that considers the taxonomical graph structure nature of GO terms in the context of an enrichment procedure that takes into account the temporal distribution of biological processes. The preliminary experimental results on the Spellman's Yeast Cell Cycle data-set encourage the use of this application of graph kernels versus a simple flat list approach based on the Jaccard distance index.

Our next concern will be to address the use of different kernel functions and dissimilarity indexes to extend the range of applicability of our approach.

References

1. Gene Ontology Consortium: The Gene Ontology (GO) project in 2006. Nucleic Acid Research (Database issue) **34** (2006) D322–D326
2. Spellman, P.T., Sherlock, G., Zhang, M.Q., Iyer, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., Futcher, B.: Comprehensive Identification of Cell Cycle-Regulated Genes of the Yeast Saccharomyces Cerevisiae by Microarray Hybridization. Molecular Biology of the Cell **9** (1998) 3273–3297
3. Li, X., Quigg, R.J.: An Integrated Strategy for the Optimization of Microarray Data Interpretation. Gene Expression **4-6** (2005) 223–230
4. Khatri, P., Draghici, S.: Ontological analysis of gene expression data: current tools, limitations and open problems. Bioinformatics **21** (2005)
5. Doherty, J.M., Carmichael, L.K., Mills, J.C.: GOurmet: a tool for Quantitative Comparison and Visualization of Gene Expression Profiles Based on Gene Ontology (GO) Distributions. BMC Bioinformatics **7**(151) (March 2006)
6. Bar-Joseph, Z.: Analyzing time series gene expression data. Bioinformatics **20**(16) (2004) 2493–2503
7. Ernst, J., Bar-Joseph, Z.: STEM: a tool for the analysis of short time series expression data. BMC Bioinformatics **7**(191) (2006)
8. Antoniotti, M., Ramakrishnan, N., Kumar, D., Spivak, M., Mishra, B.: Remembrance of Experiments Past: Analyzing Time Course Datasets to Discover Complex Temporal Invariants. Technical Report CIMS TR2005-858, Bioinformatics Group, Courant Institute of Mathematical Sciences, New York University (February 2005)
9. Ramakrishnan, N., Antoniotti, M., Mishra, B.: Reconstructing Formal Temporal Models of Cellular Events using the GO Process Ontology. In: Bio-Ontologies SIG Meeting, ISMB, Detroit MI, U.S.A. (2005)
10. Kleinberg, S., Antoniotti, M., Tadepalli, S., Ramakrishnan, N., Mishra, B.: Remembrance of Experiments Past: A Redescription Based Tool for Discovery in Complex Systems. In: Proceedings of the International Conference on Complex Systems, Boston, MA, U.S.A. (June 2006)
11. Antoniotti, M.: GOALIE site. http://bioinformatics.nyu.edu/Projects/GOALIE/ (2004-2007)
12. Ben-Hur, A., Noble, W.S.: Kernel methods for predicting protein-protein interactions. Bioinformatics **21** (2005)
13. Schölkopf, B., Tsuda, K., Vert, J.P.: Kernel Methods in Computational Biology. MIT Press (2004)
14. Cortes, C., Haffner, P., Mohri, M.: Positive Definite Rational Kernels. In: Proceedings of the 16$^{\mathrm{th}}$ Annual Conference on Learning Theory, Springer-Verlag (2003) 41–56

15. Gärtner, P., Flach, P., Wrobel, S.: On Graph Kernels: Hardness Results and Efficient Alternatives. In: COLT/Kernel. Volume 2777 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2003) 129–143
16. Kashima, H., Tsuda, K., Inokuchi, A.: Marginalized Kernels between Labelled Graphs. In: Proceedings of ICML. (2003)
17. Kondor, R.S., Lafferty, J.: Diffusion Kernels on Graphs and Other Discrete Structures. In: Proceedings of ICML. (2002)
18. Borgwardt, K.M., Cheng, S.O., Schönauer: Protein Function Prediction via Graph Kernel. Bioinformatics **21** (2005)
19. Joslyn, C.A., Mniszewski, S.M., Fulmer, A., Heaton, A.: The Gene Ontology Categorizer. Bioinformatics **20** (2004)
20. Lord, P.W., Stevens, R., Brass, A., Goble, C.A.: Investigating semantic similarity measures across the Gene Ontology: the relationship between sequence and annotation. Bioinformatics **19**(10) (2003)
21. Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press (2002)
22. Loganantharaj, R., Cheepala, S., Clifford, J.: Metric for Measuring the Effectiveness of Clustering of DNA Microarray Expression. BMC Bioinformatics **7** (2006)
23. Al-Shahrour, F., Diaz-Uriarte, R., Dopazo, J.: FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. Bioinformatics **20** (2004) 578–580
24. Beißbarth, T., Speed, T.P.: GOstat: find statistically overrepresented Gene Ontologies within a group of genes. Bioinformatics **20**(9) (2004) 1464–1465
25. Robinson, P.N., Wollstein, A., Bohme, U., Beattie, B.: Ontologizing gene-expression microarray dat: characterizing clusters with Gene Ontology. Bioinformatics **20**(6) (2004) 979–981

An Empirical Comparison of Dimensionality Reduction Methods for Classifying Gene and Protein Expression Datasets

George Lee[1], Carlos Rodriguez[2], and Anant Madabhushi[1]

[1] Rutgers, The State University of New Jersey,
Department of Biomedical Engineering,
Piscataway, NJ 08854 USA
anantm@rci.rutgers.edu
[2] University of Puerto Rico
Mayagez, PR 00681-9000

Abstract. The recent explosion in availability of gene and protein expression data for cancer detection has necessitated the development of sophisticated machine learning tools for high dimensional data analysis. Previous attempts at gene expression analysis have typically used a linear dimensionality reduction method such as Principal Components Analysis (PCA). Linear dimensionality reduction methods do not however account for the inherent nonlinearity within the data. The motivation behind this work is to demonstrate that nonlinear dimensionality reduction methods are more adept at capturing the nonlinearity within the data compared to linear methods, and hence would result in better classification and potentially aid in the visualization and identification of new data classes. Consequently, in this paper, we empirically compare the performance of 3 commonly used linear versus 3 nonlinear dimensionality reduction techniques from the perspective of (a) distinguishing objects belonging to cancer and non-cancer classes and (b) new class discovery in high dimensional gene and protein expression studies for different types of cancer. Quantitative evaluation using a support vector machine and a decision tree classifier revealed statistically significant improvement in classification accuracy by using nonlinear dimensionality reduction methods compared to linear methods.

Keywords: dimensionality reduction, bioinformatics, gene expression, proteomics, classification, prostate cancer, lung cancer, ovarian cancer, principal component analysis, linear discriminant analysis, multidimensional scaling, graph embedding, Isomap, locally linear embedding.

1 Introduction

The information found in gene and protein expression studies provides a means for identifying patients with cancer and hence these studies have emerged as promising techniques for cancer detection [1,2]. A typical gene expression dataset, however, contains information from thousands of genes (features), which are likely

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 170–181, 2007.

to be significantly greater than the number of patients from whom the data was collected. The relatively small number of patient samples compared to the very large size of the feature space results in the so-called 'curse of dimensionality' problem from a data analysis perspective [3]. Many of the genes within the expression studies may be non-informative or redundant and hence may not contribute very much from a classification perspective [4]. Two common approaches to making the data amenable to classification are (i) feature selection and (ii) dimensionality reduction (DR).

Feature selection refers to the elimination of genes determined as either being highly correlated with other genes or non-informative with respect to distinguishing the data classes [4]. It serves as a direct method for reducing inherent data dimensionality prior to classification by acquiring an optimal subset of genes to maximally separate the data classes. However, since a typical gene microarray records thousands of gene expressions, each associated with a particular gene, the cost of finding an optimal subset from several million possible combinations becomes a near intractable problem.

The alternative, dimensionality reduction (DR), is advantageous because all of the original data is simply transformed from the original high dimensional feature space to a space of eigenvectors, capable of describing the data in far fewer dimensions. The largest eigenvectors represent the direction along which the greatest variability in the dataset occurs. Advantages of DR over feature selection include (i) representation of data structure in far fewer dimensions and (ii) the visualization of individual data classes and possibly subclasses within the high dimensional data.

The most popular method for DR is Principal Components Analysis (PCA). PCA finds orthogonal eigenvectors which account for the greatest amount of variability in the data. However, its basic intuitions lie under the assumption that the data is linear. These embedded eigenvectors represent low dimensional projections of linear relationships between data points in high dimensional space. Dai et al. [5] and Shi et al. [2] have independently tested the efficacy of PCA in improving the classification of gene expression datasets. Recently, methods such as Graph Embedding [6], Isometric mapping (Isomap) [7], and Locally Linear Embedding [8] have been developed to reduce the dimensionality of nonlinear data under the assumption that the underlying distribution is nonlinear. The structure of nonlinear data can be thought of as a high order curve or manifold where the geodesic distance between two points on the manifold is greater than their Euclidean distance would suggest. Nonlinear methods attempt to map data along this nonlinear manifold by assuming only neighboring points to be similar enough to be mapped linearly with minimal error. The nonlinear manifold can then be reconstructed based on these locally linear assumptions, providing the groundwork for a nonlinear mapping based on the true distances between any two data points. In general however, the choice of DR methods for the analysis of medical data has been relatively arbitrary. Although there is widespread evidence [2,9,10] to suggest that medical data such as genomic and proteomic expression studies are nonlinear, surprisingly few researchers have attempted nonlinear DR methods for this purpose. Shi and Chen [2] have evaluated the use of LLE in comparison with PCA for improving classification in leukemia, lymphoma and colon gene expression datasets. Dawson et al. [9] explored the utility of Isomap in comparison with PCA and linear multidimensional scaling (MDS) in oligonucleotide datasets, and Nilsson et al. [10] independently compared Isomap with MDS to reveal structures in microarray data related to biological phenomena. Madabhushi et al. [6] demonstrated the use of graph embedding to detect the presence of new tissue classes on high dimensional prostate MRI studies. While significant work in comparing classifier performance on cancer studies has been done [4,11], no serious quantitative comparisons involving multiple DR algorithms have been done in the context of maximizing classification accuracy.

The primary motivation of this paper is twofold. Firstly, by quantitatively comparing the performance of multiple linear and nonlinear DR methods, we can determine the appropriate technique to precede classification in high dimensional gene and protein expression studies. Secondly, we wish to demonstrate that nonlinear DR methods are superior compared to linear methods both from the perspective of classification and from the perspective of identifying and visualizing new classes within the data. In this work, we consider genomic and proteomic expression datasets from 7 separate studies corresponding to prostate, lung and ovarian cancers, as well as leukemia and lymphoma. Three different linear methods (PCA, linear discriminant analysis (LDA) [3], linear MDS [10]) and three nonlinear DR methods (graph embedding [6], Isomap [7], and LLE [8]) are applied to each of the datasets. The low dimensional embedding vectors, obtained from each DR method and for each dataset, are then supplied to a support vector machine classifier and a decision tree classifier. The accuracy of each classifier in distinguishing between cancer and non-cancer classes is thus used to gauge the efficacy of each of the DR methods. In addition to classification, we also quantitatively compare each of the DR methods in terms of their ability to detect new sub-classes within the data.

The organization of the rest of this paper is as follows. In Section 2, we will give a brief overview of the DR methods considered in this work. In Section 3 we describe our experimental design. Our qualitative and quantitative results in comparing the different DR methods are presented in Section 4. Finally, we present our concluding remarks in Section 5.

2 Description of Dimensionality Reduction Methods

In this section we briefly describe the 3 linear (PCA, MDS, LDA) and 3 nonlinear DR methods (Graph Embedding, Isomap, LLE) considered in this study.

2.1 Linear Dimensionality Reduction Methods

Principal Components Analysis (PCA): PCA has been widely documented as an effective means for analyzing high dimensional data [5]. Briefly, PCA applies a linear transformation to the data that allows the variance within the data to be expressed in terms of orthogonal eigenvectors. The eigenvectors that contain the most variance in the data represent the *principal components*.

Linear Discriminant Analysis (LDA): LDA [3] takes into account class labels to find intra-class correlations in the dataset. Assuming there is a linear hyperplane that can maximize separation between the two classes, LDA projects features that maximally account for this inter-class difference. While LDA has been useful as both a DR method and a classifier, it is limited in handling sparse data in which a Gaussian distribution of data points does not exist [3].

Classical Multidimensional Scaling (MDS): MDS [10] is implemented as a linear method that uses Euclidean distances between each pair of points as a basis for a low dimensional data arrangement. From these input distances, MDS finds optimal positions for the data points in an arbitrary d-dimensional space by minimizing least square error. Thus, the relative Euclidean distances between points in low-dimensional embedding space are preserved. Note that classical MDS differs from nonlinear variants of MDS such as nonmetric MDS, which do not preserve input Euclidean distances.

2.2 Nonlinear Dimensionality Reduction Methods

Graph Embedding (GE): The GE algorithm [6] performs a series of normalized cuts on the data to partition it into clusters of data points. These cuts are made where minimal similarities exist (decided using a similarity matrix of pairwise Euclidean distances). In this manner, similarity can be discerned by inter- and intra-cluster distances, where points within a cluster are deemed similar and points belonging to separate clusters are deemed dissimilar. Separating points by GE allows for the separation of objects within complex nonlinear structures, where objects cannot otherwise be discriminated by linear DR methods.

Isometric Mapping (Isomap (ISO)): The Isomap algorithm [7] is essentially one that optimizes classical MDS for the nonlinear case. Isomap finds the nonlinear manifold on which the data is expected to lie through the use of a neighborhood map, which assumes linearity only between its **k** nearest neighbors defined by the user. By connecting each point only to its nearest neighbors, a path representing the geodesic distances between two points can be approximated by finding the shortest path through the neighborhood mapping. These new geodesic distances represent the true distances between points and serve as input into classical MDS, where a more accurate low dimensional representation can be constructed.

Locally Linear Embedding (LLE): LLE [8] attempts to create a low dimensional representation of the global structure through the preservation of the local structure by assuming only nearby points to be linear. Local linearity is achieved by weighting only the **k**-nearest neighbors of each data point. A total of d new embedding vectors are then reconstructed by these linear weights and by minimizing the embedding cost function in the new d-dimensional coordinate system.

3 Experimental Design

In this Section, we first briefly describe the datasets considered in this study along with a description of the parameter settings for the DR methods (Section 3.1), followed by a brief description of the classifiers considered (Section 3.2) and our model for performing a quantitative comparison of the different DR methods (Section 3.3).

3.1 Description of Datasets and Parameter Settings

To evaluate the different DR methods, we chose 7 publicly available datasets corresponding to high dimensional protein and gene expression studies[1]. The size of the datasets ranged from 34 to 253 patient samples and comprised from between 4026 to 15154 genes. Table 1 lists all the datasets on which we tested our DR methods. Note that for each dataset considered, the number of samples is significantly smaller than the dimensionality of the feature space. For the ALL-AML Leukemia dataset, two classes were considered: Acute Lymphoblastic Leukemia (ALL) and Acute Myeloid Leukemia (AML). For the DLBCL-Harvard dataset, the 2 classes considered were Diffuse large B-cell Lymphoma (DLBCL) and Follicular Lymphoma (FL). Lastly, the Lung Cancer dataset contains two types of lung cancer (mesothelioma (MPM) and adenocarcinoma (ADCA)).

Table 1. Gene expression and proteomic spectra datasets considered in this study

Dataset	*Samples*	*Genes*	*Class Description*	*Source*
(1) ALL-AML Leukemia	34	7129	20 ALL, 14 AML	Golub et al. [12]
(2) DLBCL-Harvard	77	6817	58 DLBCL,19 FL	Shipp et al. [13]
(3) Lung Cancer	148	12533	15 MPM, 134 ADCA	Gordon et al. [14]
(4) Lung Cancer-Michigan	96	7129	86 Tumor, 10 Normal	Beer et al. [15]
(5) Ovarian Cancer	253	15154	162 Tumor, 91 Normal	Petricoin et al. [16]
(6) Prostate Cancer	34	12600	25 Tumor, 9 Normal	Singh et al. [17]
(7) Types of Diffuse Large B-cell Lymphoma	47	4026	24 Germinal, 23 Activated	Alizadeh et al. [18]

For each of the datasets D_j, $1 \leq j \leq 7$, we applied each of 6 DR methods M, where $M \in \{PCA, LDA, MDS, GE, ISO, LLE\}$. For each method M and dataset D_j, we obtained a set $S^d_{D_j,M} \geq \left\{E^1_{D_j,M}, E^2_{D_j,M}, ..., E^d_{D_j,M}\right\}$ of d dominant eigenvectors. The number of principal eigenvectors d used to classify the objects $c \in D_j$ were varied from 2 to 8 in order to find the optimal d-dimensional space in which the 2 classes were most easily separable.

[1] The datasets were obtained from the Biomedical Kent-Ridge Repositories at http://sdmc.lit.org.sg/GEDatasets/Datasets and http://sdmc.i2r.a-star.edu.sg/rp

3.2 Classifiers

To perform our classification, we input a set $S^d_{D_j,M}$ of eigenvectors to the following 2 machine learning classifier methods: Support Vector Machines (SVMs) and C4.5 Decision Trees. Both require the use of a training set to construct a prediction model for new data. SVMs project the input data to a higher dimensional space to find a hyperplane that gives the greatest separation between the data classes. This hyperplane along with 2 parallel support vectors serve as a boundary in which a prediction can be made for new data. Decision Trees create a predictor wherein new samples are categorized based on several conditional statements. For each condition, the algorithm associates a certain likelihood that a sample falls into a particular category and refines the class hypothesis before a final decision is made.

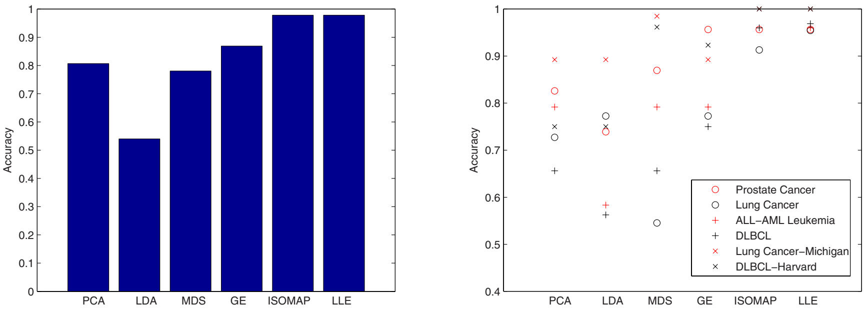
Since the classifiers were being used to evaluate the DR methods' ability to separate 2 classes, a simple linear kernel was chosen for SVM. The linear kernel draws a d-dimensional hyperplane to act as a decision boundary between the two separated classes. To train the 2 classifiers, we set aside 1/3 of the samples in each dataset $D_j, 1 \leq j \leq 7$, for 3-fold cross validation. Using the best samples from cross validation, we determined the parameter settings for both classifiers. After the model parameters were learned, the same parameter values for SVM and C4.5 were used to test the remaining 2/3 objects in each D_j.

3.3 Quantitative Evaluation of DR Methods

The accuracy of the SVM and C4.5 classifiers on 7 datasets $D_j, 1 \leq j \leq 7$ was quantitatively evaluated using the class labels provided in the gene expression studies. We define accuracy as the ratio of the number of objects $c \in D_j, 1 \leq j \leq 7$, correctly labeled by the classifier to the total number of tested objects in each D_j. We denote the classification accuracy of SVMs on dataset $S^d_{D_j,M}$ by SVM($S^d_{D_j,M}$) and the corresponding accuracy of the C4.5 Decision Trees by C4.5($S^d_{D_j,M}$). To determine whether the classifier results from the nonlinear and linear DR methods were significantly different, we performed a paired student t-test wherein we compared SVM($S^d_{D_j,M}$) for $M \in \{PCA, LDA, MDS\}$ versus SVM($S^d_{D_j,M}$) for $M \in \{GE, ISO, LLE\}$ across all D_j. The t-test was similarly repeated for C4.5($S^d_{D_j,M}$). The difference between SVM($S^d_{D_j,M}$) or C4.5($S^d_{D_j,M}$) for each pair of methods (1 linear and 1 nonlinear) was deemed to be statistically significant if $p \leq 0.05$. The linear and nonlinear DR methods were also semi-quantitatively compared (i) using 2-D embedding plots to evaluate their ability to distinguish between the cancer and non-cancer clusters and (ii) to potentially identify and visualize the presence of new sub-classes. In order to visualize the embedding of the data in the low dimensional space obtained via the DR methods, we plotted the dominant eigenvectors obtained for each M for each object $c \in D_j$ against each other (e.g. $E^2_{D_j,M}$ versus $E^3_{D_j,M}$).

4 Results and Discussion

In Section 4.1 we present the results of quantitative comparison of the different DR methods in terms of their classification accuracy obtained from the SVM and C4.5 classifiers. In Section 4.2 we present quantitative graphical plots comparing the ability of the DR methods to separate the data classes and also in identifying and visualizing the presence of new classes.



(a) (b)

Fig. 1. (a) Average C4.5($S^d_{D_j,M}$) for $d = 6$ and (b) SVM($S^d_{D_j,M}$) for $d = 5$ for each of 6 datasets following DR by *PCA*, *LDA*, *MDS*, *GE*, *ISO*, and *LLE*. Note that for both classifiers, the nonlinear methods consistently outperform the linear methods.

4.1 Quantitative Evaluation of DR Methods Via Classifier Accuracy

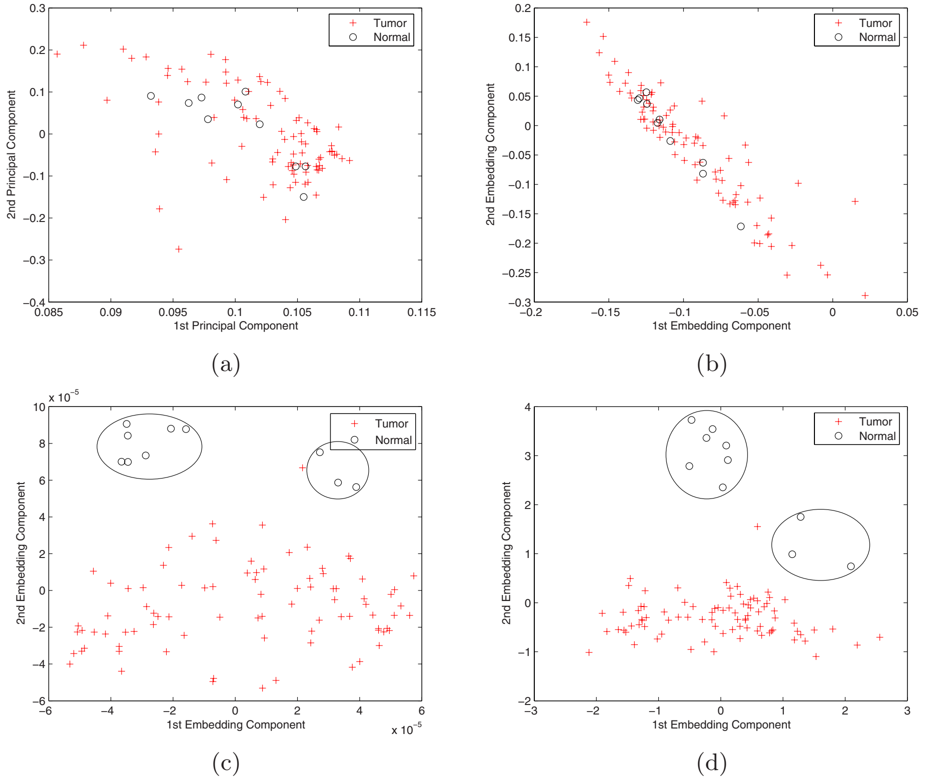
In Figure 1(a), we show the average accuracy results obtained with the C4.5 classifier, C4.5($S^d_{D_j,M}$) for $1 \leq j \leq 6$. We obtained our best results for $d =$ 5 for SVMs and $d = 6$ for C4.5 Decision Trees. From Figure 1(a), it is clear that embeddings from nonlinear DR methods (*GE*, *ISO*, *LLE*) lead to better overall accuracy than with linear DR methods (*PCA*, *LDA*, *MDS*). Isomap and LLE overall were the most accurate while LDA performed the worst. In

Table 2. C4.5($S^d_{D_j,M}$) for each of 7 datasets following DR by *PCA*, *LDA*, *MDS*, *GE*, *ISO*, and *LLE* for $d = 6$

Dataset	*PCA*	*LDA*	*MDS*	*GE*	*ISO*	*LLE*
(1) ALL-AML Leukemia	62.5	41.7	62.5	91.7	95.0	95.0
(2) DLBCL-Harvard	69.2	40.4	84.6	86.5	96.9	96.9
(3) Lung Cancer	67.7	84.6	70.8	98.5	100.0	100.0
(4) Lung Cancer-Michigan	67.7	84.6	69.2	98.5	100.0	100.0
(5) Ovarian Tumor	55.6	59.2	61.5	59.2	59.8	63.3
(6) Prostate Cancer	100.0	47.8	87.0	82.6	100.0	100.0
(7) Types of Diffuse Large B-cell Lymphoma	93.8	59.4	90.6	93.8	95.0	95.0

Table 3. p-values obtained by a paired student t-test of SVM($S^d_{D_j,M}$) across 7 data dimensions $d \in \{2,...8\}$ comparing linear versus nonlinear DR methods for $1 \leq j \leq 7$. Note that the numbers listed in the first column refer to the datasets given in Table 1.

	GE	*GE*	*GE*	*ISO*	*ISO*	*ISO*	*LLE*	*LLE*	*LLE*
Dataset	*vs*	*vs*	*vs*	*vs*	*vs*	*vs*	*vs*	*vs*	*vs*
	PCA	*LDA*	*MDS*	*PCA*	*LDA*	*MDS*	*PCA*	*LDA*	*MDS*
(1)	.068	$8x10^{-5}$	.057	$7x10^{-5}$	$6x10^{-8}$	.002	$7x10^{-5}$	$6x10^{-8}$	.002
(2)	.373	$3x10^{-4}$	.925	.012	$6x10^{-5}$	.014	.012	$6x10^{-5}$	.014
(3)	.361	.852	.691	.003	.009	$4x10^{-4}$	.002	.006	$4x10^{-4}$
(4)	.706	.063	1.000	.004	$3x10^{-8}$	.015	.005	$2x10^{-16}$	.011
(5)	.478	.063	.412	.008	.004	.003	.002	.003	$2x10^{-4}$
(6)	.156	.001	.045	$2x10^{-5}$	$4x10^{-8}$	.012	$8x10^{-5}$	$3x10^{-8}$	.019
(7)	.005	10^{-4}	.074	$8x10^{-5}$	$7x10^{-6}$	.001	$8x10^{-5}$	$7x10^{-6}$	.001



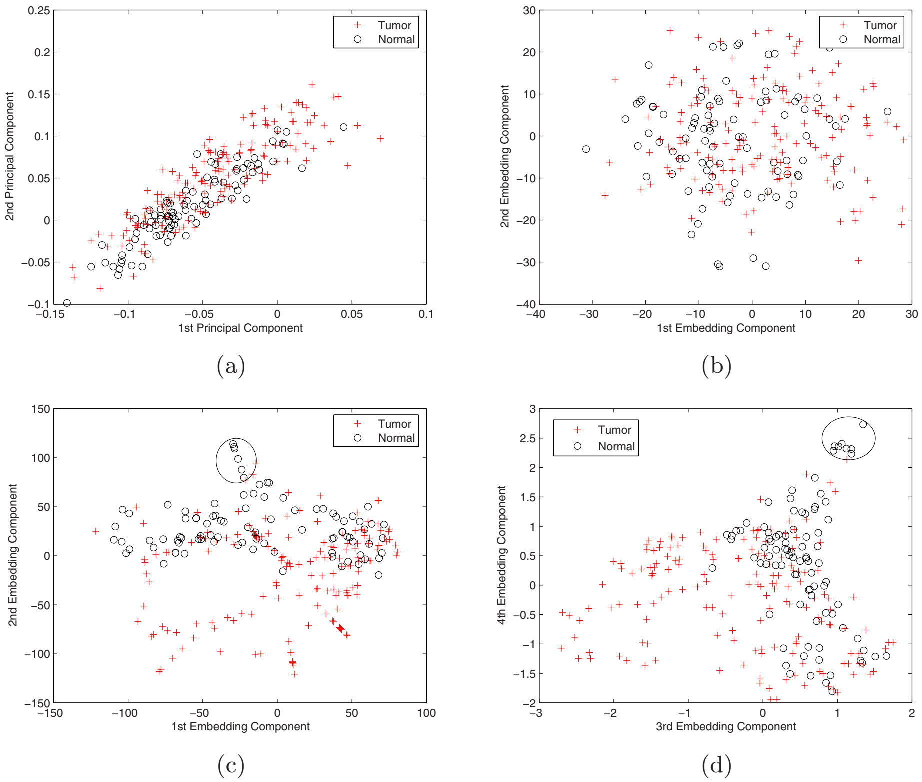
(a)

(b)

(c)

(d)

Fig. 2. Embedding plots were obtained by graphing the 2 dominant eigenvectors against each other for (a) PCA, (b) LDA, (c) GE, and (d) LLE for the Lung Cancer-Michigan dataset. Note that while linear methods, PCA and LDA, are unable to distinguish between the 2 classes, the nonlinear methods, GE and LLE, are able to not only clearly distinguish between the 2 groups but also permit visualization of 2 possible normal class sub-clusters (indicated by superposed ellipses in (c) and (d)).



(a)

(b)

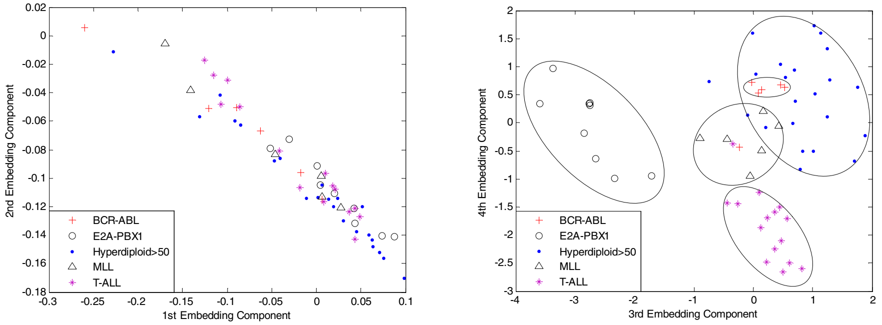
(c)

(d)

Fig. 3. Embedding plots were obtained by graphing the 2 dominant eigenvectors against each other for (a) PCA, (b) MDS, (c) Isomap, and (d) LLE for the Ovarian Cancer dataset. As in Figure 3, we can appreciate that nonlinear methods, Isomap and LLE, are able to distinguish between the 2 classes and also permit visualization of a possible normal class sub-cluster (indicated by superposed ellipses in (c) and (d)).

Table 2, we show C4.5($S^d_{D_j,M}$), for $1 \leq j \leq 7$, for all 6 DR methods, for $d = 6$. Our results indicate an improvement in accuracy for the nonlinear DR over linear DR methods. In Table 3 are listed p-values for the paired student t-tests obtained for SVM($S^d_{D_j,M}$) across 7 data dimensions ($d \in \{2, ..., 8\}$) for each paired comparison of a linear and non-linear DR method. Hence we compared the following pairs of methods: PCA/GE, LDA/GE, MDS/GE, PCA/Isomap, LDA/Isomap, MDS/Isomap, PCA/LLE, LDA/LLE, MDS/LLE for each of the 7 datasets considered. As the results in Table 3 indicate, differences in classification accuracy for pairs PCA/GE and MDS/GE were not statistically significant ($\text{p} \geq 0.05$) while all corresponding paired comparisons involving LLE and Isomap were statistically significantly more accurate compared to linear DR methods.

The results in Figure 1 and Tables 2 and 3 clearly suggest that nonlinear DR methods result in higher statistically significant accuracy compared to linear DR methods, as determined by 2 separate classifiers. Additionally, Isomap and



(a) (b)

Fig. 4. Embedding plots were obtained by graphing 2 dominant eigenvectors against each other for 5 different types of Acute Lymphoblastic Leukemia. In Figure 4(a), an embedding plot for linear LDA is compared with Figure 4(b), an embedding plot of nonlinear LLE. Note that the linear method fails to distinguish the ALL sub-classes in the reduced eigenspace, while the LLE plot clearly reveals the presence of 5 distinct clusters (indicated by superposed ellipses).

LLE were found to generate the most useful embeddings resulting in highest classification accuracy.

4.2 Semi-quantitative Evaluation of Dimensionality Reduction Methods Via Class Separation and Novel Class Detection

We evaluated the efficacy of nonlinear DR methods in identifying new data classes and intermediate cancer types. This was done by visual inspection of 2-D cluster plots obtained by plotting $E^1_{D_j,M}$ versus $E^2_{D_j,M}$ for each of the 6 DR methods. The results of the 2-D embedding plots for the Lung Cancer-Michigan and Ovarian Cancer datasets are shown in Figures 2 and 3 respectively. In Figure 2, two distinct sub-classes can be distinguished in the normal class (indicated by superposed ellipses) for GE (Figure 2(c)) and LLE (Figure 2(d)) as well as a clear, distinct separation between the cancer and non-cancer classes. For the linear embeddings obtained via PCA and LDA (shown in Figures 2(a) and (b) respectively), there appears to be significant overlap between cancer and non-cancer classes. The poor class separation is reflected in the poor classification accuracy obtained with both the SVM and C4.5 Decision Tree classifiers in Figure 1 and Table 2.

In Figure 3, we have shown the comparison of embedding plots for linear PCA (Figure 3(a)) and MDS (Figure 3(b)) against nonlinear methods, Isomap (Figure 3(c)) and LLE (Figure 3(d)), on the Ovarian Cancer dataset. In Figures 3(c) and (d), one can appreciate a sub-cluster of normal samples (indicated by superposed ellipses), possibly suggesting pre-malignant cases. More importantly, the fact that this unusual clustering is present in both nonlinear DR algorithms strongly suggests the validity of the identified sub-clusters and the utility of nonlinear DR methods in visualizing biological relationships between samples and in new class discovery.

Since we were unable to quantitatively evaluate the validity of the sub-clusters detected by the nonlinear DR methods in Figures 2 and 3, we also compared linear and nonlinear methods on a multiclass dataset. Our aim is to demonstrate that nonlinear DR methods are more capable of detecting subtypes of Acute Lymphoblastic Leukemia (ALL) [19]. Figures 4(a) and (b) show plots comparing LDA and LLE in clustering 5 subtypes of ALL. As shown in 4(a), LDA does not provide any inter-class distinction while the embedding provided by LLE in 4(b) enables easy separation between the multiple sub-classes in ALL.

5 Concluding Remarks

In this paper we presented the results of quantitatively comparing the performance of 6 different DR methods (3 linear, 3 nonlinear) from the perspective of classification and the identification of new object classes in high dimensional gene and protein expression datasets for prostate, lung, and ovarian cancers, as well as for leukemias and lymphomas. The eigenvectors obtained from each of the different DR methods were supplied to two different classifiers (SVMs and Decision Trees) to distinguish between data classes within 7 different gene and protein expression studies. Classification accuracy with both SVMs and C4.5 Decision Trees were found to be consistently higher when using features obtained by nonlinear DR methods compared to linear methods. Among the nonlinear methods, LLE gave the highest overall accuracy. In addition to distinguishing between known classes, we were also able to identify the presence of several potential sub-clusters via nonlinear DR techniques. For most datasets, all the nonlinear DR methods outperformed the corresponding linear methods, differences being statistically significant in most cases. In future work we intend to quantitatively evaluate the validity of our results on several addition datasets.

Acknowledgments. This work was made possible due to grants from the Coulter foundation (WHCF 4-29349, WHCF 4-29368), the Busch Biomedical Award and the Technology Commercialization Fund at Rutgers University.

References

1. Peng Y. A novel ensemble machine learning for robust microarray data classification. Comput Biol Med. 2006, vol.36[6], pp.553-73.
2. Shi C. and Chen L. Feature Dimension Reduction for Microarray Data Analysis Using Locally Linear Embedding. APBC 2005. pp.211-217.
3. Ye J et al. Using Uncorrelated Discriminant Analysis for Tissue Classification with Gene Expression Data. IEEE/ACM Trans. Comput. Biology Bioinform. 2004, vol.1[6], pp.181-190.
4. Tan AC and Gilbert D. Ensemble machine learning on gene expression data for cancer classification. Applied Bioinformatics. 2003, pp.65-83.

5. Dai J et al. Dimension Reduction for Classification with Gene Expression Microarray Data. Statistical Applications in Genetics and Mol Biol. 2006, vol.5[1], pp.1-15
6. Madabhushi A et al. Graph Embedding to Improve Supervised Classification and Novel Class Detection: Application to Prostate Cancer. MICCAI 2005. pp.729-737.
7. Tenenbaum JB et al. A Global Geometric Framework for Nonlinear Dimensionality Reduction. Science. 2000, vol.290, pp.2319-2322.
8. Roweis ST and Saul LK. Nonlinear Dimensionality Reduction by Local Linear Embedding. Science. 2000, vol.290, pp.2323-2326.
9. Dawson K et al. Sample phenotype clusters in high-density oligonucleotide microarray data sets are revealed using Isomap, a nonlinear algorithm. BMC Bioinformatics. 2005, vol.6, pp.195.
10. Nilsson J et al. Approximate geodesic distances reveal biologically relevant structures in microarray data. Bioinformatics. 2004, vol.20, pp.874-880.
11. Madabhushi A et al. Comparing Classification Performance of Feature Ensembles: Detecting Prostate Cancer from High Resolution MRI, Computer Vision Methods in Medical Image Analysis (In conjunction with ECCV). 2006, LNCS 4241, pp. 25-36.
12. Golub TR et al. Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. Science. 1999, vol.286, pp.531-537.
13. Shipp MA et al. Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nat Med. 2002, vol.8, pp. 68-74.
14. Gordon GJ et al. Translation of microarray data into clinically relevant cancer diagnostic tests using gene expression ratios in lung cancer and mesothelioma. Cancer Res. 2002, vol.62, pp.4963-4967.
15. Beer D et al. Gene-expression Profiles Predict Survival of Patients with Lung Adenocarcinoma. Nature Medicine. 2002, vol.8[8], pp.816-823.
16. Petricoin EF et al. Use of proteomic patterns in serum to identify ovarian cancer. The Lancet. 2002, vol.359[9306], pp.572-577
17. Singh D et al. Gene expression correlates of clinical prostate cancer behavior. Cancer Cell. 2002, vol.1, pp.203-209.
18. Alizadeh AA et al. Distinct types of diffuse large B-cell lymphoma identified by gene expression profiling. Nature. 2000, vol.403, pp.503-511.
19. Yeoh EJ, et al. Classification, Subtype Discovery, and Prediction of Outcome in Pediatric Acute Lymphoblastic Leukemia by Gene Expression Profiling. Cancer Cell. 2002, vol.1[2], pp.133-143.

NEURONgrid: A Toolkit for Generating Parameter-Space Maps Using NEURON in a Grid Environment

Robert J. Calin-Jageman[1,*], Chao Xie[2,*], Yi Pan[2], Art Vandenberg[3],
and Paul S. Katz[1]

[1] Department of Biology
[2] Department of Computer Science
[3] Information Systems and Technology
Georgia State University, Atlanta, GA 30303, USA
rcalinjageman@gsu.edu

Abstract. Neuroscience research increasingly involves the exploration of computational models of neurons and neural networks. To ensure systematic model exploration, it is often desirable to conduct a parameter-space analysis in which the behavior of the model is catalogued over a very large range of parameter permutations. Here we report the development and testing of a toolkit called NEURONgrid for conducting this type of analysis in a grid environment using NEURON (Hines & Carnevale, 1997, 2001), a popular and powerful simulation platform for the neurosciences. NEURONgrid provides helper classes within NEURON for manipulating parameters, a package of NEURON for running in a grid environment, and a management client that enables neuroscientists to submit a parameter-space analysis, monitor progress, and download results. NEURONgrid provides a user-friendly means for conducting intensive model exploration within the neurosciences. It is available for download at http://neurongrid.homeip.net.

1 Introduction

Computational modeling has become an important technique within the neurosciences (Stern and Travis, 2006). Neuroscientists now regularly construct models of neurons and neural networks to integrate experimental data, test hypotheses, and generate novel predictions. Although considerable effort is spent in developing and tuning these models, the end goal is *exploration*—varying conditions and inputs to understand the model's behavior and how it might relate to the real system. Recently, there has been a drive to make model exploration more systematic by generating detailed parameter space maps (Foster et al., 1993; Goldman et al., 2001; Prinz et al., 2003, 2004). In this approach, the behavior of a model is catalogued across a relatively large set of parameter permutations. The resulting data set represents a high-dimensional parameter-space map, which is useful for tuning models to new data sets, analyzing the influence of different parameters, and assessing the uniqueness of different solution sets (see below).

[*] Co first-authors.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 182–191, 2007.

Although parameter-space mapping is a useful approach to model exploration, it is a combinatorial task that requires extensive computational power for expansive mapping of a parameter-space. Currently, only two published solutions are available for computational neuroscientists. Prinz et al. (2003) wrote customized code for distributing a neural network simulation on a Beowulf cluster. However, this solution only scales to the size of the cluster and requires significant development effort. Another approach, NeuronPM (Calin-Jageman & Katz, 2006), produces a screen-saver cluster in which the screen-saver client embeds NEURON (Hines & Carnevale, 1997, 2001), a common simulation environment for neurons and neural networks. This second approach reduces development time by utilizing an existing simulation platform that can run existing models without modification. However, the screen-saver client in NeuronPM lacks a scripted install/configuration routine, cannot be updated automatically, and is not capable of encrypted communication with the server. This approach is thus most suitable for installation on local intranets of relatively modest size (e.g. university computer lab).

To overcome the deficiencies of the existing approaches, we have developed a toolkit, called NEURONgrid, for generating parameter-space maps with NEURON in the GridMP environment (United Devices, 2003). NEURONgrid includes helper classes in NEURON for varying parameters in existing models, a custom package of NEURON for use within GridMP, and a user-friendly management client the can submit large parameter sweeps, monitor progress, and download result sets. NEURONgrid provides a stable, rapid, and scalable solution for parameter-space mapping that will work with existing NEURON models. Below we describe a) the benefits of parameter-space mapping, b) the GridMP platform and its utility for this type of analysis, c) the components of NEURONgrid, and d) the results of a test case. NEURONgrid is open source and available at http://neurongrid.homeip.net

2 Uses of Parameter-Space Maps in Neuroscience

Model neurons and neural networks commonly involve tens to hundreds of parameters. Thus, probing any significant part of a model's parameter space will necessarily involve millions of simulations runs. Given the extreme computational cost, it is worth considering in more detail why this type of analysis is desirable.

Tuning models. One of the hardest tasks in computational neuroscience is identifying a parameter set for a model that will produce a desired behavior. A parameter-space map can provide a useful reference set for tuning a model (Prinz et al., 2003). In this approach, the parameter-space map becomes a kind of look-up table for selecting model configurations and/or seeding tuning algorithms. This may be of limited utility for models with an unwieldy number of parameters (Achard & De Schutter, 2006), but it has proven useful for identifying useful parameter sets for both individual neurons and neural networks (Prinz et al., 2004).

Stability analysis. When a model neuron or network exhibits an interesting behavior, it is important to examine the range of conditions under which the behavior can be maintained. A parameter-space map can identify the necessary and sufficient conditions for maintaining a behavior. Moreover, it can identify the sensitivity of the behavior to different model parameters and their multivariate interactions (e.g.

Foster et al., 1993; Goldman et al., 2001; Achard & De Schutter, 2006). This can provide fundamental insight into the mechanisms underlying the behavior of interest. It may also prove useful for simplifying complex models, as only a subset of total model parameters are likely to exert significant control on model behavior (Tobin et al., 2006) and some neural processes can have redundant effects (Calin-Jageman & Fischer, 2003).

Range of model behaviors. Neurons and neural networks often exhibit multiple output modes. For example, some neurons can switch between tonic spiking and bursting (e.g. Beurrier et al., 2000). Parameter-space mapping provides a technique for sampling the output modes of a model and elucidating the inputs that can switch between the modes (Calin-Jageman et al., 2006).

3 GridMP Environment

Parameter-space mapping is computationally intensive, but it requires only the simplest form of parallelization—each point in the parameter space is mapped as an independent run of the simulation without any need for resource or data sharing across runs. Thus, simple 'task-farming' (Gonzalez-Velez, 2005) is a very relevant paradigm for speeding the creation of a parameter-space map. In this respect, the Grid MetaProcessor (GridMP) environment is ideal as it uses cycle harvesting to distribute tasks among many autonomous computers (United Devices, Austin, TX). Further, GridMP provides full encryption of code and data (input and output). This section provides an overview of the GridMP platform architecture by discussing data flow and the major GridMP platform components (United Devices, 2003).

Figure 1 shows the major components in the GridMP platform and identifies which components communicate with each other.

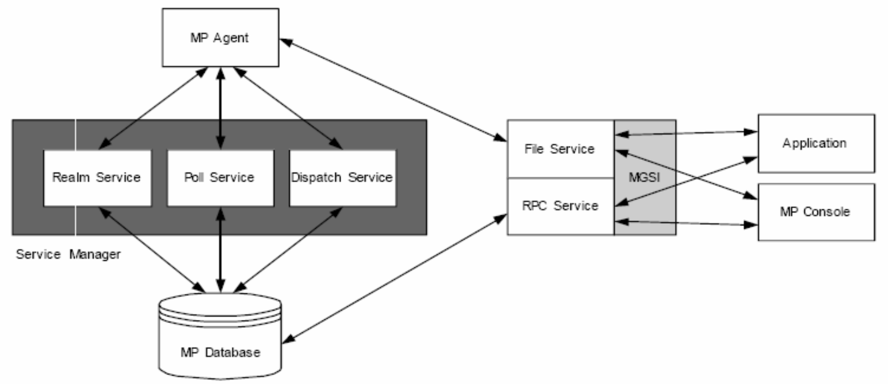


Fig. 1. The GridMP platform components and data flow

The workflow in the GridMP platform is described as follows:

1. The system administrator starts the GridMP platform Service Manager.
2. Devices (computers in the GridMP domain) connect to the Realm Service for authentication and to receive credentials.

3. Devices contact the Dispatch Service and work, if available, is assigned to each device.
4. The MP Agent running on each device downloads any required files from the File Service and begins processing the work.
5. While processing the work, MP Agents report usage statistics to the Poll Service.
6. When the work is finished, the MP Agents upload Results to the File Service and request more work from the Dispatch Service.
7. The MP Database is a repository of system state and metadata for GridMP.

Figure 2 shows the MP Services, the components with which they communicate, and the communication protocol(s) each component uses.

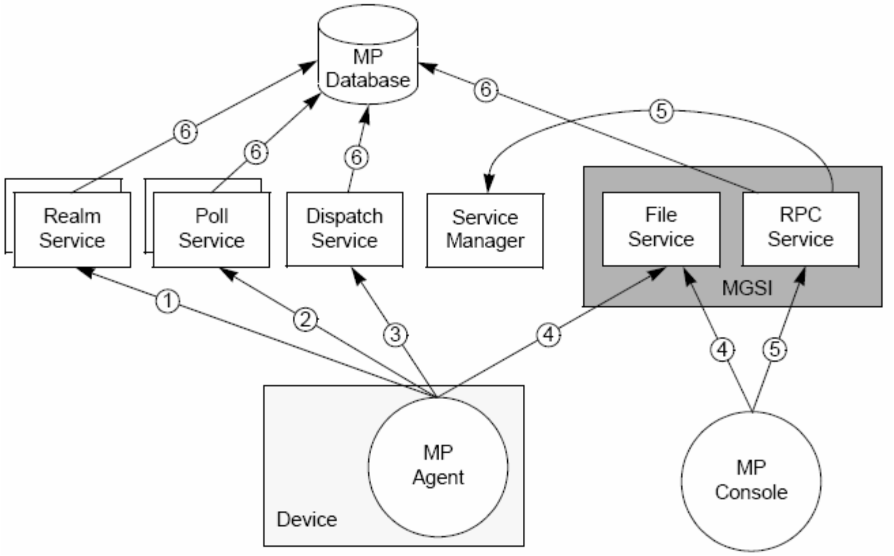


Fig. 2. Communication and transport protocols between platform components

The GridMP platform is comprised by the following components, MP database, MP agent, service manager, realm service, poll service, dispatch service, file service, and program loader.

The MP Database is the main storage repository of common information for the GridMP platform. The MP Database stores system state and metadata; the File Service server stores Job and Application data.

The MP Agent is a lightweight program that runs on a device and manages Job processing. The MP Agent is responsible for processing work (Program Module executable data), and for automatically returning Result files to the GridMP platform.

The primary function of the Service Manager is to ensure that all required services needed for the proper functioning of the GridMP platform are always up and running. If any are found to have failed or unexpectedly stopped, the Service Manager restarts the failed service.

The Realm Service manages all MP Agent authentication and MP Agent resource access. MP Agents communicate with the Realm Service to register and authenticate themselves. An MP Agent cannot perform any operation until it has authenticated itself with the Realm Service.

The MP Poll Service collects periodic status reports from MP Agents and communicates commands to the MP Agent from other services.

The Dispatch Service schedules Workunits to devices based on device capabilities and availability, and receives Result status from those devices. Devices that are idle in the GridMP platform connect to the Dispatch Service to receive new Workunits. The Dispatch Service selects and sends a Workunit to each connecting device.

The Dispatch Service includes a workload scheduler that schedules Workunits to devices based on Job and device preferences and attributes. When a Workunit is dispatched, all metadata about it is also sent. The files containing actual data are uploaded or downloaded through the File Service.

The File Service is a secure (SSL-encrypted) file transmission service for downloading and uploading data to and from MP Agents during Job execution. The File Service relies on an underlying file system. MP Agents can also use the GridMP platform URL mechanism to download data from a list of specified URLs.

The Program Loader is a mandatory loader that loads executable code into memory for execution, and provides automatic encryption and compression for the executables.

4 Toolkit

For a simulation environment, we selected NEURON (Hines & Carnevale, 1997, 2001). NEURON is robust, open source, cross-platform, extensible, and popular. It provides a C-like scripting library with built-in classes for specifying and running model neurons and networks. New modeling formalisms can be incorporated using a model description language. A graphical interface is also available and is sufficient for complete model development and testing. Versions are available for Linux, Windows, and OS X, and models are completely portable across these systems. Attesting to the popularity of NEURON within the neurosciences, the most recent annual meeting of the Society for Neuroscience (http://www.sfn.org/) featured over 50 presentations of research conducted with NEURON.

We developed NEURONgrid as a toolkit to enable NEURON to generate parameter-space maps within the GridMP environment. This involved the development of classes in NEURON for manipulating parameter values, packaging NEURON to run on a GridMP client, and developing a management client to enable neuroscientists to quickly deploy a parameter-space analysis.

4.1 Helper Classes for Parameter Manipulation

We wrote a set of helper classes using the NEURON scripting language to provide a standard interface for manipulating parameters. A parameter is defined with the name of the real model variable and a list or function for varying that variable's values. To set up a parameter-space analysis, users create a simple text file that defines each parameter to be varied. A control class provides all the functionality required to execute the parameter-space search. This class loads the existing NEURON model, reads the parameter list, and retrieves work assignments from the GridMP manager. Work assignments are delivered as a decimal code representing a point in parameter space and an integer value for the number of points in space to explore before returning results. The control class sets the model to the corresponding point in parameter space, runs the simulation, collects results, and then moves the model forward to the next point in the search. These control classes enable a parameter-space analysis of an existing NEURON model by simply defining a text file of parameters—no further modification or setup is needed.

4.2 NEURON Package for GridMP

Executables that run on the GridMP platform must use GridMP platform tools, software, and protocols to perform file management and task execution. We used the buildmodule tool from the GridMP development toolkit to create a program module for NEURON suitable to run on grid clients. This tool bundles the target program with a program Loader and Module Definition File (MDF), creating a Program Module Executable. The Program Loader creates the execution environment for the executable. The MDF describes how to run the executable and describes the available data packages and their encoding. Prior to building the package, we stripped out extraneous files (documentation and demos) from the standard NEURON installation. This ensured a clean and compact Program Module for running NEURON on grid clients.

We also needed a way to detect the status of NEURON running on a grid client. We thus developed a small monitor program and bundled it inside the program module. The monitor program polls the status of the NEURON program. When the desired set of simulation runs has completed, the monitor program notifies the GridMP server.

4.3 Management Client

Finally, we developed a management client to enable neuroscientists to submit large parameter sweeps, monitor progress, and download result sets. The management client was developed in Visual C#. It interacts with GridMP via webservices.

The management client is meant to run locally on the machine of the neuroscientist creating the parameter-space analysis. The client enables the neuroscientist to identify a model directory containing a complete NEURON model, supporting files, and a text file with parameter definitions. The management client splits the analysis up into distinct workunits, each consisting of a starting point in the parameter space and the number of adjacent points to map. The model is then uploaded and the batch of workunits is submitted to the MP server.

After deployment, the client monitors the status of the submitted jobs on the GridMP server. It reports back the percentage of completed workunits and refreshes automatically every minute. When the parameter-space map is complete, the management client collects results from all workunits and downloads them to the local machine for analysis. Each results file is automatically numbered to correspond to its workunit, making it easy to script an import routine for database or analysis software.

5 Test Case

5.1 Experimental Setup

To test NEURONgrid, we replicated a large-scale parameter space analysis conducted by Calin-Jageman et al. (2006). The analysis was performed on a four neuron model of the Tritonia swim neural network (Frost et al., 1997). This model network uses a hybrid integrate-and-fire scheme that incorporates some realistic ionic conductances (Getting, 1989). In the real neural system, the network is largely quiescent but produces bouts of rhythmic neural activity when the organism senses a predator. Calin-Jageman et al. (2006) conducted a parameter-space analysis of this model to understand the determinants of the quiescent and rhythmic states of the network. In their analysis, the weights of 9 synaptic connections were varied, each taking on 5 different values. Thus, their analysis included 5^9 (~1.9 million) runs, each simulating a 90s time period with a 1ms time step.

Running this simulation with NEURONgrid required no modification of the existing NEURON model. We simply defined the analysis in a text file and uploaded the model files using the management client. We then used the management client to deploy the analysis on a GridMP environment at Georgia State University. This grid consisted of 573 nodes around the GSU campus connected on a 100MB Ethernet. Within the grid, there were 329 Windows-based clients. As we have not yet packaged OSX and Linux versions of NEURON, only these 329 machines were available.

To test the performance of NeuronGrid, we deployed a test parameter-space analysis with different work-pool sizes, ranging from a single machine to 300 machines. Runs were completed in triplicate and the performance values averaged. This was done to smooth out small fluctuations in run-time due to heterogeneity in the quality of available clients. For most runs, we did not complete the entire analysis of ~1.9 million runs, but selected a subsection (1-3%) of the parameter-space and scaled performance results accordingly. We did, however, test several complete runs of the parameter-space utilizing the full client pool.

5.2 Results

We found that running the analysis on a single computer would have required ~63.9 days of continuous processing time. Distributing this task into a pool of 64 work units, however, required ~1 day of processing time. Thus, we observed that runtime decreased roughly as a factor of the size of the work pool. In the log-log plot of work time versus pool size shown in Figure 3, this appears as a straight line with a negative slope.

For each pool size tested, we calculated the parallel speedup (run time on a single machine divided by the run time for that pool size). This data is shown in Figure 4 and shows an approximately linear speedup with the size of the work pool. For

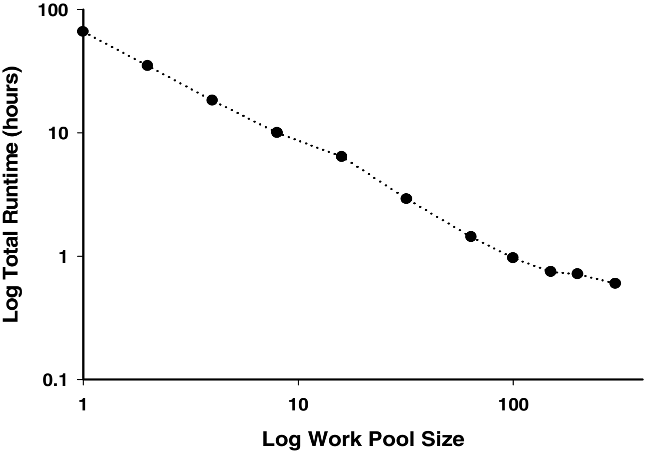


Fig. 3. Average runtime (hours) required for NEURONgrid to complete the sample parameter-space analysis plotted by the size of the work pool deployed. Units shown in log scale.

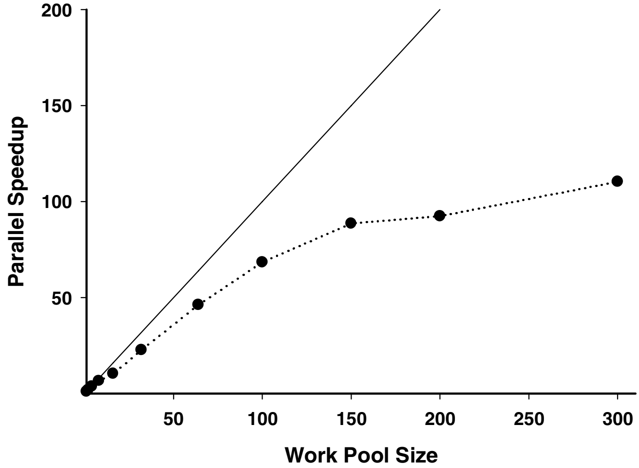


Fig. 4. Average speedup on the sample parameter-space analyis by the size of the work pool deployed. Speedup was calculated as the ratio of single-unit runtime to the runtime for that work-pool size. For reference, the solid line shows a unitary line (slope of 1) representing the ideal maximum speedup.

reference, a unit line is also shown (solid) representing the ideal maximum speedup. It can be seen that NEURONgrid deviates from this ideal as the work pool size increases over 32 units. Moreover, performance begins to plateau with work pool sizes over 100 units. The initial deviation is due two factors. First, there was substantial heterogeneity amongst the grid clients. The single-computer run used to calculate speedup represents the fastest-performing clients. As work pool size expands and clients of lesser power are utilized, performance fails to reach ideal speedup. A related factor is that increased work pools increase the chances of a grid client falling offline (reboot, network error, etc).

The plateau in performance comes as the client pool saturates. Although the GSU grid had 329 grid clients available, this plateau becomes evident with a work pool size as small as 100 units. This is because the default behavior of GridMP is to assign each work unit to multiple clients. It may be possible to enhance performance by decreasing assignment redundancy, but this would also lead to more assignment failures. Thus, the net gain may be negligible.

We confirmed these results with several complete runs of the sample parameter-space analysis. As predicted by our partial runs, we were able to generate a complete data set of 1.9 million runs in less than 0.6 days (<14 hours). This yielded 360MB of summary data for download with the management client. For comparison, NeuronPM completed the same analysis in 4 days with a client pool of 16 computers. This indicates similar speedup for both systems. NeuronPM, however, would be difficult to maintain with a client-pool size over 30. Thus, GridMP offers the opportunity for significantly better performance because of its ability to scale to many hundreds of clients.

The rapid speed of parameter-space mapping in a grid environment suggests that this type of analysis can become more typical within the neurosciences. Moreover, significantly larger sections of parameter space could be mapped, especially with access to larger grid systems, such as the World Community Grid (http://www.worldcommunitygrid.org/) with hundred of thousands of systems.

6 Discussion and Future Work

Computational modeling is growing increasingly prominent within the neurosciences. As the use of modeling grows, so does the need for effective tools for exploring and analyzing these models. The toolkit, NEURONgrid, which we have developed, provides a user-friendly means for generating large-scale parameter-space maps in a grid environment. Although a handful of other solutions are available for this type of effort, NEURONgrid offers a number of advantages. First, it is built around the established NEURON simulation platform, and existing models can be adapted for the grid with little or no modification. Second, it utilizes the GridMP environment, which is scalable, secure, and available in many university contexts and commercial environments. Finally, it is user-friendly. The helper classes in NEURON and management client should enable neuroscientists to easily develop and deploy very complex parameter-space analyses with a minimum of effort. The 2007 meeting of the Computational Neuroscience Society will feature a special workshop on utilizing parallel computing with NEURON (http://www.neuron.yale.edu). This attests to the growing popularity of this approach and the continued need to develop tools for supporting such large-scale analyses.

Future work will be done packaging NEURON for Linux and OS X clients in GridMP, enabling the full client pool to be utilized. In addition, we are exploring ways to archive simulation results in a database for sharing and long-term reference.

References

Achard, P., & De Schutter, E. (2006). Complex parameter landscape for a complex neuron model. *PLoS Computational Biology*, *2*(7): e94.

Beurrier, C., Bioulac, B. & Hammond, C. (2000). Slowly inactivating sodium current (I(NaP)) underlies single-spike activity in rat subthalamic neurons. *Journal of Neurophysiology*, *83*, 1951-1957.

Calin-Jageman, R.J. & Fischer, T.M. (2003). Synaptic augmentation contributes to environment-driven regulation of the *Aplysia* siphon-withdrawal reflex. *Journal of Neuroscience,* 23: 11611-11620

Calin-Jageman, R.J., Frost, W.N. & Katz, P.S. (2006). Neuromodulatory control of rhythmic neural activity in the Tritonia swim CPG: a large-scale computational analysis. *Annual Meeting of the Society for Neuroscience*, 32: 350.1.

Calin-Jageman, R.J. & Katz, P.S. (2006). A distributed computing tool for generating neural simulation databases. *Neural computation.* 18: 2923-2927.

Foster, W.R., Ungar, L.H. & Schwaber, J.S. (1993). Significance of conductances in Hodgkin-Huxley models. *Journal of Neurophysiology*, 70, 2502-2518.

Frost, W.N., Lieb, J.R., Tunstall, M.J., Mensh, B. & Katz, P.S. (1997). Integrate-and-fire simulations of two molluscan neural circuits. In: *Neurons, networks and motor behavior*, Stein, P. ed., MIT Press. Cambridge. pp. 173-179.

Getting, P.A**.** (1989). A network oscillator underlying swimming in *Tritonia*. In: *Neuronal and Cellular Oscillators*, edited by J. W. Jacklet, New York:Marcel Dekker, Inc, p. 215-236.

Goldman, M.S., Golowasch, J., Marder, E. & Abbott, L.F. (2001). Global structure, robustness, and modulation of neuronal models. *Journal of Neuroscience,* 21: 5229-5238.

Gonzalez-Velez, H. (2005). An adaptive skeletal task farm for grids. In *Euro-Par 2005 Parallel Processing*, *Lecture Notes in Computer Science* 3648, 401-410.

Hines, M.L. & Carnevale, N.T. (1997). The NEURON simulation environment. *Neural Computation,* 9:1179-1209.

Hines, M.L. & Carnevale, N.T. (2001). NEURON: a tool for neuroscientists. *Neuroscientist* 7: 123-135.

Prinz, A.A., Billimoria, C.P. & Marder, E. (2003). Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of Neurophysiology,* 90:, 3998-4015.

Prinz, A.A., Bucher, D. & Marder, E. (2004). Similar network activity from disparate circuit parameters. *Nature Neuroscience,* 7: 1345-1352.

Stern, P. & Travis, J. (2006). Of bytes and brains. *Science*, 314: 75.

Tobin, A-E., Van Hooser, S.D. & Calabrese, R.L. (2006). Creation and reduction of a morphologically detailed model of a leech heart interneuron. *Journal of Neurophysiology*, *96*: 2107-2120.

United Devices (2003). GridMP Platform: architecture overview. Online at www.ud.com/resources/files/br_gridmp.pdf

An Adaptive Resolution Tree Visualization of Large Influenza Virus Sequence Datasets

Leonid Zaslavsky, Yiming Bao, and Tatiana A. Tatusova

National Center for Biotechnology Information,
National Library of Medicine, National Institutes of Health,
8600 Rockville Pike, Bethesda, MD 20894, USA
{zaslavsk,bao,tatiana}@ncbi.nlm.nih.gov
http://www.ncbi.nlm.nih.gov

Abstract. Rapid growth of the amount of influenza genome sequence data requires enhancing exploratory analysis tools. Results of the preliminary analysis should be represented in an easy-to-comprehend form and allow convenient manipulation of the data.

We developed an adaptive approach to visualization of large sequence datasets on the web. A dataset is presented in an aggregated tree form with special representation of sub-scale details. The representation is calculated from the full phylogenetic tree and the amount of available screen space. Metadata, such as distribution over seasons or geographic locations, are aggregated/refined consistently with the tree. The user can interactively request further refinement or aggregation for different parts of the tree.

The technique is implemented in Javascript on client site. It is a part of the new AJAX-based implementation of the NCBI Influenza Virus Resource.

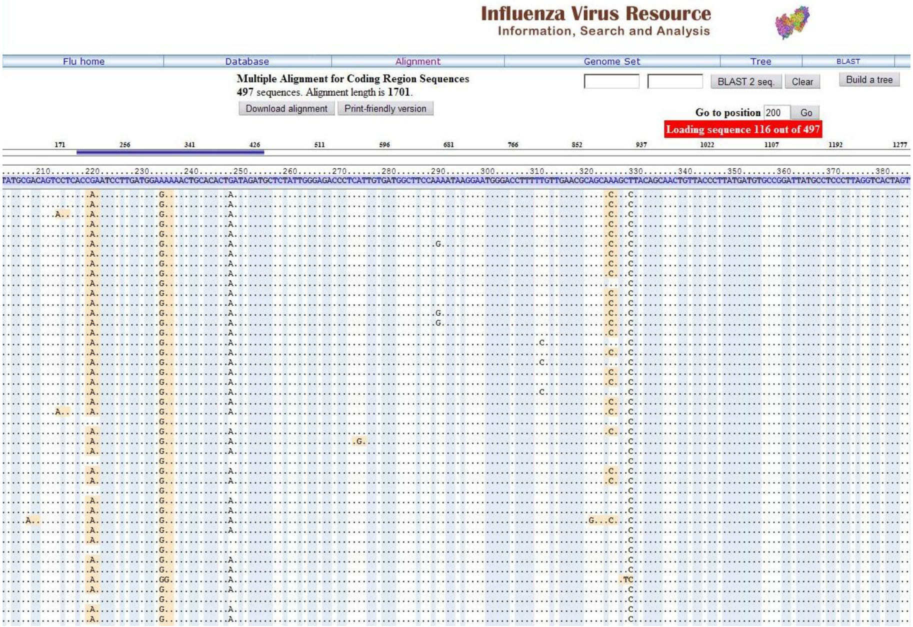
Keywords: visualization, adaptive, sequence, tree, phylogenetic, virus, influenza, JavaScript, AJAX.

1 Introduction

The number of influenza virus sequences in the public database more than doubled from the beginning of 2005, thanks to collaborative genome sequencing efforts by the National Institute of Allergy and Infectious Diseases ([1], [2]), St. Jude Children's Research Hospital, the Centers for Disease Control and Prevention, and many others. This requires more sophisticated preliminary analysis tools to be provided to users. Datasets should be represented in an easily comprehensible and adjustable visual form that provides a convenient way of manipulating the data.

The visualization approaches used in several releases of the NCBI Influenza Virus Resource ([3],[4]) were based on sequence-level representation of the data. They provided a convenient interface for viewing the entire dataset and manipulating individual sequences: viewing multiple sequence alignments and trees built

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 192–202, 2007.



Multiple Alignment for Coding Region Sequences
497 sequences. Alignment length is 1701.

Loading sequence 116 out of 497

Fig. 1. Multiple sequence alignment

using different algorithms [5] (typical web visualizations of a multiple sequence alignment and a phylogenetic tree are shown in Figures 1, 2). However, the approach based on manipulating individual sequences is not very useful for large datasets. For example, detailed schematic representation of a huge dataset with a fine level of detail, with all information included regardless of relevance, is very difficult to comprehend ([6], [7]). There are many influenza virus sequences that are identical or highly similar to each other. Most of the time, it is not necessary to show all such sequences in the analysis. Also, operating the data manually sequence-by-sequence is highly inefficient and time-consuming for the user. In addition, the user needs guidance to scan through a complex set of data provided not only at the level of individual sequences but also groups of sequences, depending on the task. It is preferable to structure the dataset and provide meaningful aggregated representations with the ability to adapt the aggregation level.

Several systems have been developed to support interactive browsing of large trees with the ability to focus ([8], [9], [10], [11]). The issues of scalability, performance and robustness of tree visualization have been also addressed [12]. In addition, innovative approaches to visualization of geographic information have been developed [13]. Modern cartographical systems widely used in mobile devices provide adaptively-coarsened visual representations of maps. Such representation changes in real time to provide the best visualization suiting a specific task (driving, flying). In each of these cases, the information helpful for performing

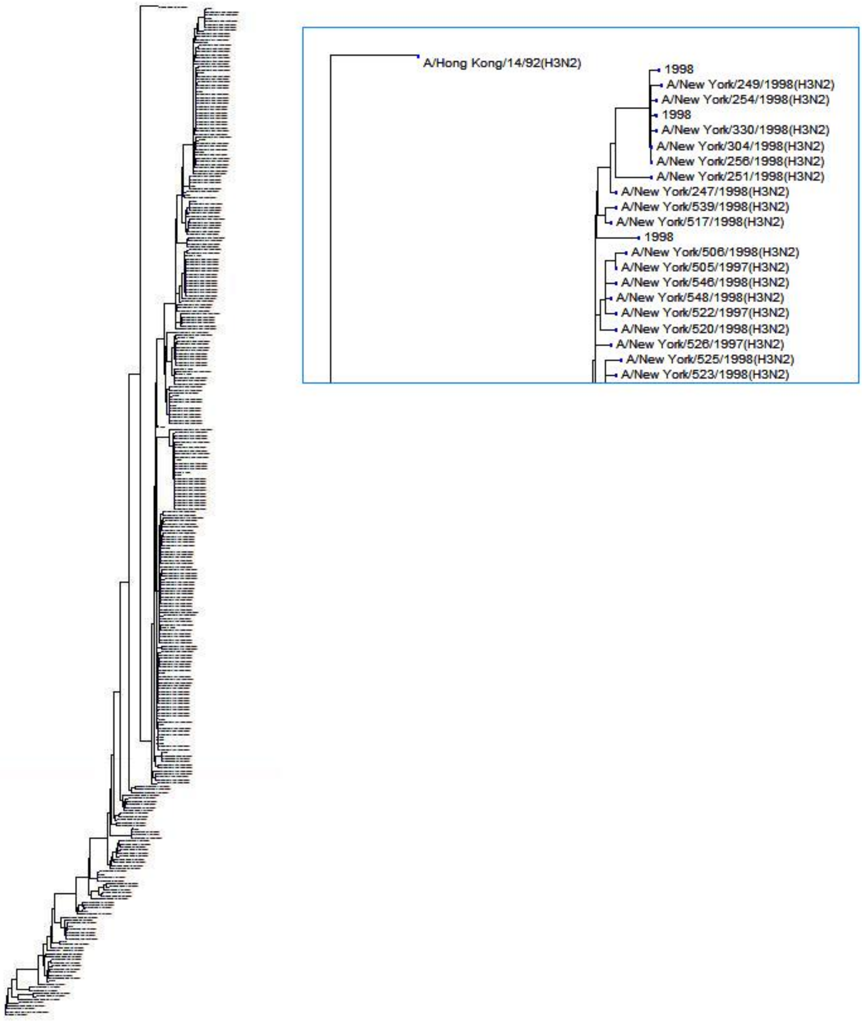

A/Hong Kong/14/92(H3N2)
1998
A/New York/249/1998(H3N2)
A/New York/254/1998(H3N2)
1998
A/New York/330/1998(H3N2)
A/New York/304/1998(H3N2)
A/New York/256/1998(H3N2)
A/New York/251/1998(H3N2)
A/New York/247/1998(H3N2)
A/New York/539/1998(H3N2)
A/New York/517/1998(H3N2)
1998
A/New York/506/1998(H3N2)
A/New York/505/1997(H3N2)
A/New York/546/1998(H3N2)
A/New York/548/1998(H3N2)
A/New York/522/1997(H3N2)
A/New York/520/1998(H3N2)
A/New York/526/1997(H3N2)
A/New York/525/1998(H3N2)
A/New York/523/1998(H3N2)

Fig. 2. An full-resolution tree built for 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during a 20-year period (1968-1998), using the neighbor-joining method. The top of the tree is enlarged in the small window.

the task is provided. The knowledge is represented in an easy-to-comprehend form and the amount of information is limited in a way that a human (driver) can process it and make a reasonable decision in real time.

We propose an adaptive approach to visualize the dataset in an aggregated form adapted to the user's screen, allowing the user to interactively refine or aggregate visualization of different parts of the dataset, depending on the task and need for details. The essential parts of our technique are:

- Representation of a large tree by a smaller tree having aggregated groups as terminal nodes;
- Placing a specially constructed tree to show the structure of each aggregated group at the sub-scale resolution level.
- Creating metadata description for each aggregated group from the original metadata.

Sub-scale resolution representation. When a tree for aggregated group is calculated, it can be shown as a phylogenetic tree with groups shown as named terminal nodes. However, this representation can be refined within the same screen space. Since the height of the font used in annotation is usually several pixels (typically, 10-12 px), available vertical space can be used to show, in some form, the structure of the subtree corresponding to the aggregated group.

An example of aggregated tree is shown in Figure 3 (the dataset is the same as in Figure 2).

Initial tree visualization, built from the full tree taking into account available screen space, can be changed by the user interactively through requesting refinement for some aggregated groups and further aggregating subtrees that are not of interest to the user. The user can also change the tree annotation

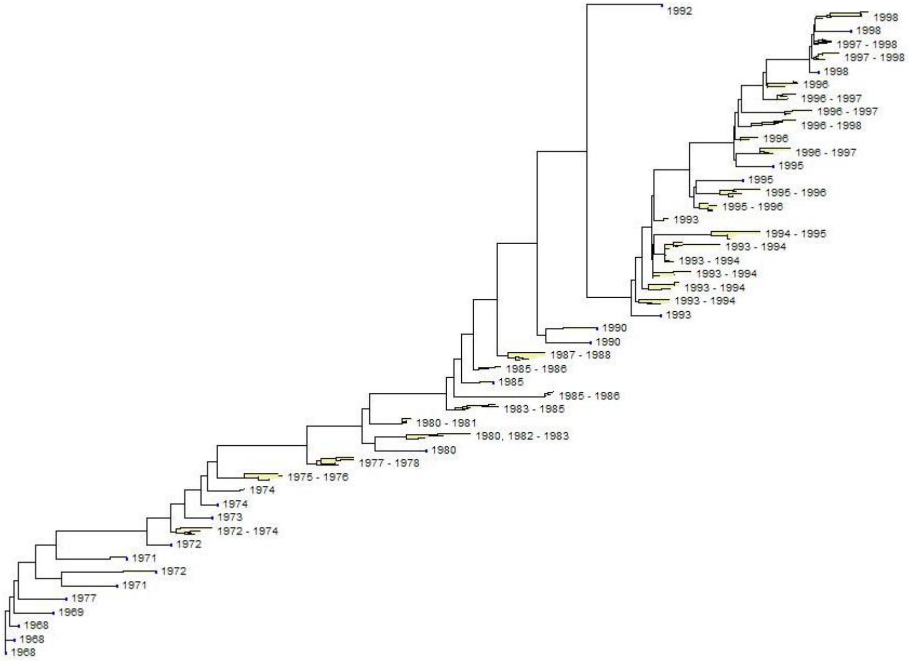


Fig. 3. An aggregated tree built for 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during 20-year period 1968-1998 (the full tree was calculated using the neighbor-joining method). The dataset is the same as in Figure 2.

by indicating special interest for particular influenza, subtypes, years, seasons or geographic locations. This will cause sequences of interest to be recolored at the sequence-level representation, and annotation of aggregated groups to be changed and recolored, to reflect information requested by the user.

While full tree is built on the server, its adaptive visualization is built and can be interactively changed on the client-machine using a JavaScript implementation. It is embedded in the new version of our analysis tools based on the AJAX technology [14]. Since we specifically aim work within a browser on client machine, we are limited to graphic functionality available in HTML[1].

2 Methods

Data Structures. The tree is implemented using an array of nodes, with each node containing array indexes of its parent node and children. We defined the following Javascript objects:

`Tree` - object containing an array of `Node` objects as its `nodes` property, root index in the node array as its `rootId` property, and some auxiliary objects. In the process of adaptive aggregation, object `SubtreeInfo` is added.

`Node` - an object having the following properties:

- `parentId` - an index of the parent node in the array `nodes`, or `-1` for root;
- `children` - an array containing indices of children nodes in `nodes` array;
- `branchLength` - length of the branch going to the parent node;
- `metadata` - node metadata (name, subtype, date of extraction, country, etc.);
- `presentation` - information on visual representation;
- `status` - an integer value, initially equal to `0`. Value `1` is set of of corresponding subtree should be represented in the aggregated form.

Object `SubtreeInfo` has the following fields:

`lengthMin` - minimal distance from the root of the subtree to a leaf;
`lengthMax` - maximal distance from a leaf of the subtree to a leaf;
`diam` - diameter of the subtree.

Distances used in calculations of `lengthMin`, `lengthMin`, and `lengthMin` are tree distances, i.e. lengths of shortest paths.

Calculating subtree information for original tree. `SubtreeInfo` objects, containing minimal and maximal distances from the subtree root to a leaf and

[1] Non-linear two-dimensional transformations requiring rich graphical functionality are the core of the approaches like [10].

diameter of the subtree are calculated for tree nodes in a bottom-to-top manner using formulas:

$$l_i^{min} = \min\{l_j + l_j^{min} | j \in \Omega_i\}; \tag{1}$$

$$l_i^{max} = \max\{l_j + l_j^{max} | j \in \Omega_i\}; \tag{2}$$

$$d_i = \max\left(\max\{d_j | j \in \Omega_i\}, \max\{d_j + d_k + l_j + j_k | j, k \in \Omega_i, j \neq k\}\right); \tag{3}$$

where

Ω_i is the subtree having node i as its root,
l_i is the branch length from node i to its parent,
l_i^{min} is minimal tree distance from node i to a leaf in subtree Ω_i,
l_i^{max} is maximal tree distance from node i to a leaf in subtree Ω_i,
d_i is diameter of subtree Ω_i.

Building an aggregated tree. In order to control visualization, integer *status* variable `Tree.nodes[i].status` is assigned to each node i of the full tree. It has the following meaning:

If $status = 0$, the node is treated as a usual tree node.
If $status = 1$, the node is treated as an aggregated group:

- The name of the aggregated group is created and shown;
- The aggregated group is shown graphically using sub-scale visualization.

In the beginning, we assign root *status* to *1*, i.e. consider the tree as aggregated in one group. We start disaggregate nodes, from the child of the root that has largest diameter of the corresponding subtree, and disaggregate while screen space allows (Technically, desegregating node i means setting its *status* to *0* and setting *status* of its children to *1*).

To control the order of node disaggregation, we use auxiliary array Θ, where indices of the candidate nodes for disaggregation sorted by non-increasing diameters are placed:

$$d_{i_k} \geq d_{i_m} \text{ for any } k < m, 0 \leq k, m < |\Theta| \tag{4}$$

Technically, array Θ is included in the `Tree` object as `Tree.leafArray.nodeIds`. It is easy to see that

$$d_{i_0} = \max\{d_{i_k} \mid 0 \leq k < |\Theta|\} \tag{5}$$

i.e., the node in the front of the array has the maximal diameter of the subtree.

The disaggregation algorithm is described as follows. The number of groups in the aggregated tree is denoted as N, the maximal allowed number of groups as N_{max}, and the set of children of node i as Λ_i.

Algorithm 1.

```
Set root $status$ to $1$;
Include root in $\Theta$;
Set $N$ to $1$.
While( $|\Theta| > 0$ and $N + \max(|\Lambda_{i_0}| - 1,\ 0) \leq N_{max}$){
    Set $status$ of node $i_0$ to $0$;
    Delete $i_0$ from $\Theta$;
    If( $\Lambda_{i_0} \neq \emptyset$ ){
        For ( all $k \in \Lambda_{i_0}$ ){
            Include $k$ in $\Theta$;
            Set $status$ of node $k$ to $1$;
        }
        Set $N \leftarrow N + |\Lambda_{i_0}| - 1$.
    }
}
```

The new indices k are included in Θ with order preserved[2] (4).

Drawing sub-scale resolution representation. We represent a subtree on sub-resolution level (e.g., in the space approximately equal to the font height) as follows:

- Start from a tree containing only one element, corresponding to the root of the sub-tree and perform several steps of disaggregation;
- Represent each non-resolved subtree by two leaves: closest to the root and most distant (see Fig. 4).

Figure 5 illustrates transformation of a subtree in its sub-scale resolution representation. The algorithm for building a subscale-resolution tree is similar to Algorithm 1.

Aggregating Metadata. When aggregated groups of sequences are created, we can create abstracted description of the group to annotate the tree. We can summarize the group using the following descriptive characteristics:

- type;
- subtype,
- year of extraction;
- season of extraction;
- geographical location (country, continent).

However, abstracting or summarizing less formal descriptions, such as strain name, seems to be more challenging.

[2] In our current implementation, a binary search is performed to find insertion position and JavaScript method `Array::splice` is used for inserting an element in the JavaScript array.

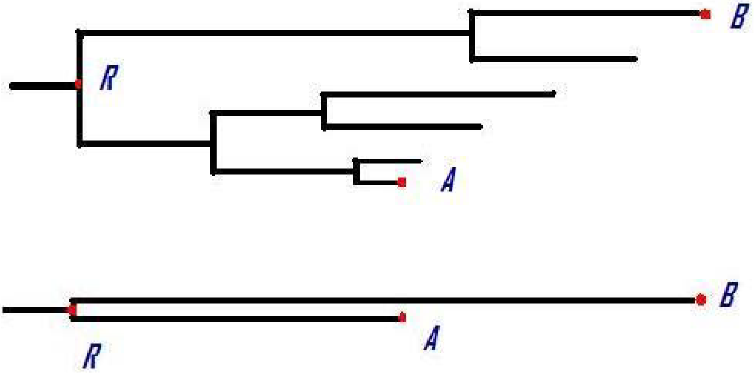


Fig. 4. Representation of of an unresolved subtree by a tree with two leaves, showing the leaf closest to the root and the leaf most distant from the root, in the original subtree

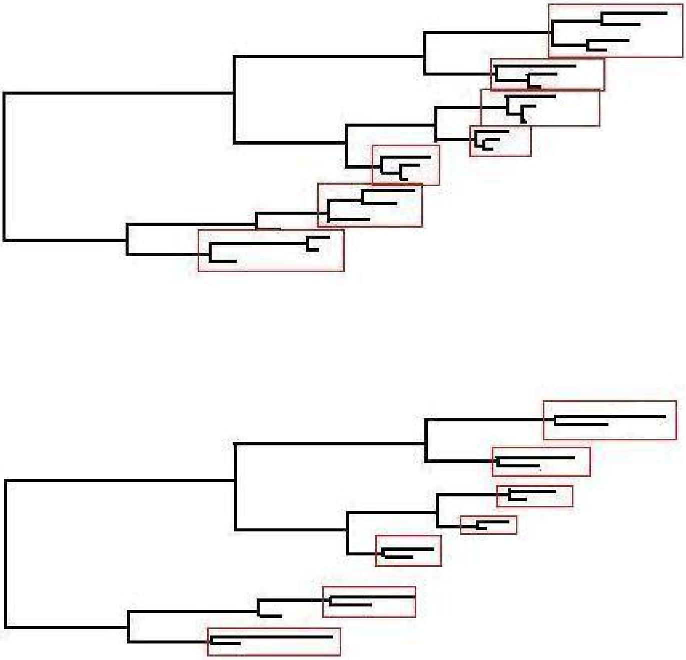


Fig. 5. A subtree (top) and its sub-scale resolution representation (bottom)

JavaScript Implementation. The JavaScript library implementing the adaptive visualization of the tree consists of two layers. The first contains objects and methods to calculate the tree for aggregated groups and trees for sub-level resolution, as well as metadata. The second contains objects and methods for actual rendering; it creates and changes HTML objects and places them to DOM tree. Separation of the logical and rendering levels facilitates easy change of rendering when necessary, the logic and data flow are completely abstracted from rendering technology.

Currently, we implement tree rendering using HTML ⟨`div`⟩ objects. In the future, however, we may decide to take a different approach to web rendering using standard non-proprietary tools. One potential candidate is the new HTML element ⟨`canvas`⟩ [15], and another one is Scalable Vector Graphics (SVG) [16]. However, neither ⟨`canvas`⟩ element, nor SVG, have become widely used standard tools providing implementation-independent output yet[3],[4].

While implementation of adaptive tree visualization purely within HTML using JavaScript allows manipulation of the tree on the client machine, and this approach has obvious advantages, it also has a drawback: the implementation of the tree using HTML elements does not allow saving the tree as an image and quality of printing depends solely on the browser functionality. In the future, we plan to provide an image for the tree: the selection made by the user within the web tool will be transferred to the server and an image in one of standard formats will be created and sent to the client.

Test Results. We applied developed methodology to typical influenza virus sequence datasets. An aggregated tree obtained for dataset containing 380 HA protein sequences for Influenza A H3N2 viruses extracted from human hosts during 20-year period 1968-1998, is shown in Figure 3 (for comparison, see Figure 2 showing the same dataset with a traditional sequence-level resolution).

3 Discussion

Adaptive aggregative visualization of datasets with the possibility to refine and coarse different parts of the representation interactively on the web is a promising approach to a convenient preliminary analysis of large datasets. It allows the user to view and manipulate the data hierarchically, doing each operation at the appropriate resolution level. We implemented this approach for tree visualization and demonstrated its efficiency and usefulness.

It is highly desirable to apply hierarchical visualization and adaptive resolution to other types of data representation. One of the immediate areas requiring our attention is multiple sequence alignment visualization. While we provide a convenient multiple alignment view in the current system (such as shown in Figure 1), the user would not be able to comprehend data at sequence level for

[3] New HTML5 element ⟨`canvas`⟩ is a part of the proposed HTML5 standard, but it is not yet implemented as a native object in all browsers. Moreover, its current limited implementation in selected browsers does not allow to include text [15].

[4] SVG requires either a native implementation within a browser or a plug-in. Partial native implementations are available in selected browsers [17]. Several plug-in implementations are available. However, Adobe Systems, the provider of Adobe SVG Viewer, stated that they will discontinue support for Adobe SVG Viewer by the end of 2007 [18].

large datasets consisting of hundreds or even thousands of sequences. A different alignment representation, that allows to adjust the resolution and focus, seems to be helpful.

Acknowledgements

This research was supported by the Intramural Research Program of the NIH, National Library of Medicine.

The NCBI Influenza Virus Resource is developed and being continuously improved as a part of a collaborative effort, the Influenza Genome Sequencing Project led by NIAID ([19], [1], [2]). The authors are thankful to David J. Lipman, Yury Voronov, Boris Fedorov, Stacy Ciufo and Sergey Ponomarev for productive discussions.

References

1. Fauci, A.S.: Race against time. Nature **435**(7041) (May 2005) 423–424
2. Ghedin, E., Sengamalay, N.A., Shumway, M., Zaborsky, J., Feldblyum, T., Subbu, V., Spiro, D.J., Sitz, J., Koo, H., Bolotov, P., Dernovoy, D., Tatusova, T., Bao, Y., St George, K., Taylor, J., Lipman, D.J., Fraser, C.M., Taubenberger, J.K., Salzberg, S.L.: Large-scale sequencing of human influenza reveals the dynamic nature of viral genome evolution. Nature **437**(7062) (October 2005) 1162–1166
3. The National Center for Biotechnology Information (NIH/NLM/NCBI): The Influenza Virus Resource. `http://www.ncbi.nlm.nih.gov/genomes/FLU/FLU.html`
4. Bao, Y., Bolotov, P., Dernovoy, D., Kiryutin, B., Zaslavsky, L., Tatusova, T.A., Ostell, J., Lipman, D.J.: NCBI Influenza Virus Resource. Manuscript in preparation. National Center for Biotechnology Information (2006)
5. Felsenstein, J.: Inferring Phylogenies. 1 edn. Cambridge University Press (September 2003)
6. Mather, G.: Foundations of Perception. 1 edn. Psychology Press (January 2006)
7. Baron, J.: Thinking and Deciding. 3 edn. Cambridge University Press (December 2000)
8. Card S. K., N.D.: Degree-of-interest trees: A component of an attention-reactive user interface. In: Proc. Advanced Visual Interfaces (AVI). (2002) 231–245
9. Fekete J.-D., P.C.: Interactive information visualization of a million items. In: Proc. InfoVis. (2002) 117–124
10. Lamping J., Rao R., P.P.: Focus+content technique based on hyperbolic geometry for viewing large hierarchies. In: Proc. CHI'95. (1995) 401–408
11. Rost U., B.B.E.: Treewiz: interactive exploration of huge trees. Bioinformatics **18**(1) (2002) 109–114
12. Beermann, D., Munznerz, T., Humphreysy, G.: Scalable, robust visualization of very large trees. In K.W. Brodlie, D.J. Duke, K.I.J., ed.: EUROGRAPHICS - IEEE VGTC Symposium on Visualization. (2005) 1–8
13. MacEachren, A.M.: How Maps Work: Representation, Visualization, and Design. 2nd revised edn. The Guilford Press (June 2004)
14. Zakas, N.C., McPeak, J., Fawcett, J.: Professional Ajax. 1 edn. Wrox (February 2006)

15. Mozilla Foundation: Resources related to the new HTML5 ⟨canvas⟩ element. http://developer.mozilla.org/en/docs/Category:HTML:Canvas
16. The World Wide Web Consortium (W3C): Scalable vector graphics (svg): Xml graphics for the web. http://www.w3.org/Graphics/SVG/
17. Mozilla Foundation: Mozilla svg project. http://www.mozilla.org/projects/svg/
18. Adobe Systems: Adobe SVG Viewer. http://www.adobe.com/svg/viewer/install/
19. The National Institute of Allergy and Infectious Diseases (NIAID): NIAID Launches Influenza Genome Sequencing Project. Press Release. http://www3.niaid.nih.gov/news/newsreleases/2004/flugenome.htm (November 2004)

Wavelet Image Interpolation (WII): A Wavelet-Based Approach to Enhancement of Digital Mammography Images

Gordana Derado[1], F. DuBois Bowman[1], Rajan Patel[1,2], Mary Newell[3], and Brani Vidakovic[1,4]

[1] Department of Biostatistics, Emory University, Atlanta, GA
[2] Amgen Inc., Thousand Oaks, CA
[3] Department of Radiology, Emory University, Atlanta, GA
[4] Georgia Institute of Technology, Atlanta, GA

Abstract. Cancer detection using mammography focuses, in part, on characteristics of tiny microcalcifications, including the number, size, and spatial arrangement of the microcalcifications, as well as morphological features of individual microcalcifications. We have developed state-of-the-art wavelet-based methods to enhance the resolution of microcalcifications visible on digital mammograms, aimed at improving the specificity of breast cancer diagnoses. In our research, we develop, refine, and evaluate a Wavelet Image Interpolation (WII) procedure and create accompanying software to implement it. WII involves the application of an inverse wavelet transformation to a coarse or degraded image and constructed detail coefficients to produce an enhanced higher resolution image. The construction of detail coefficients is supervised by the observed image and innate regular scaling assessed by a statistical model. We found that our proposed procedure is efficient and useful in capturing relevant clinical information in the context of digital mammographic imaging. Our proposed methodology was tested by an experienced radiologist using 40 images from the University of South Florida Digital Database for Screening Mammography (DDSM).

1 Introduction

In the United States, breast cancer is the second leading cause of death in women. One out of eight women will develop breast cancer in her lifetime. Studies have indicated that early detection and treatment improve the chances of survival for breast cancer patients (Curpen *et al.* [6], Smart *et al.* [20]). At present, mammography is the only proven method that can reduce breast cancer mortality through early detection. Attempts to increase the specificity of mammmographic diagnoses, and therefore to reduce the number of unnecessary biopsies, are based on the evaluation of the number, size, and spatial arrangement of microcalcifications (Millis *et al.* [18]) as well as morphological features of single microcalcifications (Egan *et al.* [8]). We propose the development and application of novel

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 203–214, 2007.

wavelet-based methods for improving cancer diagnoses by enhancing key features of microcalcifications in digital mammograms.

Although there are several findings on a mammogram image that are critical for a cancer diagnosis (masses, architectural distortions, asymmetry), our research focuses on microcalcifications. About half of the cancers detected by mammography appear as a cluster of microcalcifications. Microcalcifications are the most common mammographic sign of ductal carcinoma in situ (DCIS), which is an early stage cancer confined to the breast ducts. Almost 90% of DCIS cases are associated with microcalcifications.

In addition to specific arrangement or distribution, the irregularity of microcalcification shapes is an important attribute. “Pleomorphic” or “heterogeneous” are synonyms for an irregular shape or variability of shapes which can indicate DCIS. Such microcalcifications are usually more conspicuous than the amorphous forms (Coakley and van Doorn [5]). Another malignant form includes fine, linear or fine, linear, branching calcifications. These are thin, irregular calcifications that appear linear, but are discontinuous and under 0.5 mm in width. Their appearance suggests filling of the lumen of a duct involved irregularly by breast cancer.

The main goal of this research was to generate a procedure for enhancing the digital mammographic images, based on *wavelet transform methods*.

1.1 Brief Overview of the Discrete Wavelet Transformation (DWT)

Let $\mathbf{y}$ be a data-vector of dimension (size) n. For simplicity, we choose n to be a power of 2, say 2^J.

Suppose that we apply a wavelet transform to the vector $\mathbf{y}$ yielding a transformed vector $\mathbf{d}$ that has the same length as $\mathbf{y}$. This linear and orthogonal transform can be fully described by an $n \times n$ orthogonal matrix $\mathbf{W}$. In practice, one performs the DWT without exhibiting the matrix $\mathbf{W}$ explicitly, but by using fast filtering algorithms. The filtering procedures are based on so-called quadrature mirror filters which are uniquely determined by the wavelet of choice and fast Mallat's algorithm (Mallat [15]). The wavelet decomposition of the vector $\mathbf{y}$ can be written as

$$\mathbf{d} = (\mathcal{H}^{\ell}\mathbf{y}, \mathcal{G}\mathcal{H}^{\ell-1}\mathbf{y}, \ldots, \mathcal{G}\mathcal{H}^2\mathbf{y}, \mathcal{G}\mathcal{H}\mathbf{y}, \mathcal{G}\mathbf{y}). \tag{1}$$

where ℓ is any fixed number between 1 and $J = \log_2 n$, and the operators $\mathcal{H}$ and $\mathcal{G}$ are defined coordinate-wise by

$$(\mathcal{H}a)_k = \sum_{m \in \mathbf{Z}} h_{m-2k} a_m, \text{ and } (\mathcal{G}a)_k = \sum_{m \in \mathbf{Z}} g_{m-2k} a_m, \quad k \in \mathbf{Z}$$

where g and h are high- and low-pass filters corresponding to the wavelet of choice. Components of g and h are connected via the quadrature mirror relationship $g_n = (-1)^n h_{1-n}$. For all commonly used wavelet bases, the taps of filters g and h are readily available in the literature or in standard wavelet software packages.

The elements of $\mathbf{d}$ are called *wavelet coefficients*. The sub-vectors described in (1) correspond to detail levels in a levelwise organized decomposition. For instance, the vector $\mathcal{G}\mathbf{y}$ contains $n/2$ coefficients representing the finest level of detail.

In general, jth detail level in the wavelet decomposition of $\mathbf{y}$ contains 2^j elements, and is given as

$$\mathcal{G}\mathcal{H}^{J-j-1}\mathbf{y} = (d_{j,0}, d_{j,1}, \ldots, d_{j,2^j-1}). \tag{2}$$

Wavelet transformations of 2-D objects (images) are performed by applying univariate transformations on the rows and on the columns of the 2-D object. One step of the decomposing algorithm is described next. Consider a digital image A, which is in fact a matrix comprised of pixel values. A wavelet decomposition begins by applying the wavelet low pass filter $\mathcal{H}$ and high pass filter $\mathcal{G}$ to the rows of the matrix A. This step produces two matrices $\mathcal{H}_r A$ and $\mathcal{G}_r A$, both of dimension $2^n \times 2^{n-1}$ (the subscripts r suggest that the filters are applied on rows). Next, the filters $\mathcal{H}$ and $\mathcal{G}$ are applied to the columns of the matrices $\mathcal{H}_r A$ and $\mathcal{G}_r A$ obtained from step one, producing matrices $\mathcal{H}_c\mathcal{H}_r A, \mathcal{G}_c\mathcal{H}_r A, \mathcal{H}_c\mathcal{G}_r A$ and $\mathcal{G}_c\mathcal{G}_r A$, each of dimension $2^{n-1} \times 2^{n-1}$. The matrix $\mathcal{H}_c\mathcal{H}_r A$ is an average or smooth representation of the original image, while the matrices $\mathcal{G}_c\mathcal{H}_r A, \mathcal{H}_c\mathcal{G}_r A$ and $\mathcal{G}_c\mathcal{G}_r A$ contain details lost by degrading A to $\mathcal{H}_c\mathcal{H}_r A$. The transform is further carried out by repeating the process on the *average* matrix $\mathcal{H}_c\mathcal{H}_r A$ in place of A. In the rest of the paper we denote $\mathcal{H}_c\mathcal{H}_r A$ simply by $\mathcal{H}\mathcal{H}$, $\mathcal{H}_c\mathcal{G}_r A$ by $\mathcal{H}\mathcal{G}$ and the other matrices correspondingly.

1.2 Previous Work

Many computerized methods for detecting clustered microcalcifications based on wavelets have been proposed. Yoshida *et al.* [25] used a combined difference-image technique and a wavelet transform to detect subtle microcalcifications. First, the difference-image technique is used to increase the signal-to-noise ratio of microcalcifications, and then the wavelet-based scheme is applied to detect the subtle microcalcifications missed by the first step. To extract small-scale structures, wavelet transform (WT) uses a fine "probe" that is represented by a small wave.

Lado *et al.* [13] employed a computerized scheme for detecting "both individual microcalcifications in regions of interest (ROIs) and clustered microcalcifications over the complete mammograms, based on the application of two different wavelet transform techniques"(one-dimensional and two-dimensional WT). Wang *et al.* [24], proposed an approach for detecting microcalcifications in digital mammograms employing wavelet-based subband image decomposition. In Anastasio *et al.* [2], the wavelet transform is employed as a preprocessing step whose goal is to enhance the microcalcifications and suppress the background structure in the mammogram. A parallel-genetic algorithm is used in performing the optimization of the CAD procedure.

Strickland *et al.* [21] developed a 2-stage method based on wavelet transforms for the detection and segmentation of calcifications. The first stage is based on an nondecimated wavelet transform.

In the second stage, detected pixel sites in $\mathcal{GG}$ and $\mathcal{HG} + \mathcal{GH}$ are dilated and then weighted before computing the inverse wavelet transform (here $\mathcal{GG}, \mathcal{HG}$ and $\mathcal{GH}$ denote the detail components in the second level of decomposition). Bruce and Adhami [3] apply the discrete wavelet transform mod-max method to the problem of mammographic mass classification. This method was used to extract multiresolution features that quantify the mass shapes. They showed that when utilizing a statistical classification system with Euclidian distance measures in determining class membership, the use of multiresolution features significantly increases the classification accuracy. Ferrari *et al.* [9] presented a method for the identification of pectoral muscle in MLO (mediolateral oblique) mammograms based on the multiresolution technique using Gabor wavelets. Chang *et al.* [4] developed an enhancement algorithm relying on multiscale wavelet analysis, and extracted oriented information at each scale of analysis was investigated. Another approach to image enhancement of digital mammography images was introduced by Seršić and Lončarić [19]. It consists of three steps: low-frequency tissue density component removal, noise filtering, and microcalcification enhancement. An overview of automatic methods for detection of microcalcifications was given in a recent publication by Thangavel *et al.* [22]. To our knowledge, none of the previous methods utilized the scaling property of the mammography images to obtain the subpixel enhancement.

2 Description of the Data

The collection of images we analyzed was obtained from the University of South Florida's Digital Database for Screening Mammography (DDSM)(Heat *et al.* [10]). (See `http://marathon.csee.usf.edu/Mammography/Database.html`).

The DDSM is described in detail in Heat *et al.* [11]. Images containing suspicious areas have associated pixel-level "ground truth" information about the locations and types of suspicious regions. We selected a set of cases(studies) from the DDSM from volumes 6 and 7. Each case contains four mammograms (two for each breast, the craniocaudal (CC) and mediolateral oblique (MLO) projections) from a screening exam. We analyzed the data from 10 benign cases and 8 malignant cases, each containing calcifications.

The images were scanned on either a HOWTEK 960 or HOWTEK Multi-RAD 850 digitizer with a sample rate of 43.5 microns per second at 12 bits per pixel. They were stored in a format using lossless JPEG compression. However, even with the compression, each image file is quite large because the films were scanned with resolution between 42 and 100 microns. The source code for the program used to compress, as well as the program used to uncompress the images are available to download from the web site.

3 Methodology

Wavelets have been applied for image enhancement since the early 1990's, and some of the prime applications are in digital mammography (Aldroubi and

Unser [1], Heinlein *et al.* [12], Lemaur *et al.* [14], McLeod *et al.* [16], Wang and Karayiannis [24], etc.) Most wavelet-based approaches involve thresholding, a procedure that eliminates background noise. Wavelets are also applied to generate "difference" images in which nonessential background is eliminated. The method proposed in this paper is novel and involves inverse wavelet transforms of low-resolution images. In what follows, we present a conceptual description of the WII procedure and demonstrate the utility of this approach for digital mammography.

Our approach consists of several steps. First, an image enhancement method is used to detect the *regions of interest* on each image. The area(s) with ROIs are then cropped from the original image. (From now on we will refer to those cropped images as the "original images.") To confirm that the correct regions were identified, we compared our ROIs with the information available from the DDSM where abnormalities were marked by experienced radiologists. After detecting the regions of interest on images from our data set, a three-level wavelet decomposition was applied to each image. In order to construct informative detail spaces, the WII procedure is combined with a *linear regression* approach, based on pixel intensity *scaling*, after which thresholding is applied on the resulting images. Finally, the 2-D inverse wavelet transform was applied using the original image as the smooth part and the estimated informative details to obtain a higher resolution enhanced image of microcalcifications.

We now describe each of these steps in more detail.

3.1 Detection of Regions of Interest (ROI)

We first consider an algorithm similar to that presented in Seršić and Lončarić [19], which aims to detect microcalcifications between $0.1mm$ and $1mm$ in diameter. Thus, in an original digital mammogram of $50\mu m \times 50\mu m$ resolution, a microcalcification may appear to be 2 to 20 pixels wide. A 5-level redundant 2D wavelet decomposition on the original mammogram yields detail coefficients with a spatial resolution from $0.1mm \times 0.1mm$ to $1.6mm \times 1.6mm$, thus encompassing the range of microcalcification size considered.

Wavelet coefficient images corresponding to the first level detail coefficients seem to consist primarily of spatially white noise, in the absence of microcalcifications between $0.05mm$ and $0.1mm$ in diameter. However, in the detail coefficients of levels $2-5$, the mammogram signal becomes apparent and microcalcifications begin to visually emerge from the background noise. The approximation coefficients at the 5^{th} level contain the low frequency mammogram information such as tissue density and breast shape information.

Several different wavelet families are considered for the ROI detection algorithm, but 2D polyharmonic B-spline wavelets with quincunx subsampling (see Van de Ville *et al.* [23]) are utilized due to their symmetry and lack of directional (horizontal, vertical, and diagonal) bias in the detection of microcalcifications. A significant difference between the algorithm presented here and that given in [19] is the use of a 2-D non-separable polyharmonic wavelet basis which introduces no directional bias due to the quincunx based subsampling scheme. The algorithm presented by [19] utilizes recursive 1-D decompositions that favor horizontal, vertical, and diagonal directions, which may bias the structure of the microcalcifications.

3.2 Wavelet Image Interpolation Procedure

The proposed WII procedure is conceptually simple. If the forward wavelet transform, described previously, degrades an image by decomposition into a smooth part and the details, then the inverse wavelet transform performed on a degraded image will interpolate the image and reveal a higher degree of details. Operationally, the procedure proceeds as follows:

1. One starts with an empty image (all entries 0) and performs k wavelet decomposition steps. Of course, the transform is linear and the resulting smooth and detail sub-matrices are all zero-matrices.
2. The degraded image from a digital mammogram is inserted into the position of the smooth matrix containing zeros. This step requires that the degraded input image and the original smooth part have equal dimensions. The detail matrices from step 1 retain the 0 values in all entries.
3. Back-transform the object by k steps.

This process increases the resolution of the degraded, pixelized image and contains 4^k times the number of pixels in the original input For example, a three-step transform produces an enhanced image with 64 times the number of pixels in the original degraded image. The main contribution of our algorithm is building detail spaces based on degraded image and general scaling properties of natural images.

3.3 Imputing Details

In the WII procedure described above, the three detail matrices, accompanying the pixelized image located in the smooth part, consist entirely of zero. It is natural to propose the utilization of detail spaces to further enhance the information in the interpolated image. Several avenues are possible, including template details, background details, wavelet-bootstrap by resampling the details, etc. Our proposal is to utilize the self-similarity of wavelet decompositions in building informative detail spaces.

Most of the natural images *scale* and this scaling can be assessed in the wavelet domain. Informally, scaling means that the "energy" (squared wavelet coefficients) cascades when the resolution of wavelet decomposition changes. This is particularly true for some medical images (tissue, bones, cancer, etc). This scaling was described and utilized in statistical inference by many researchers, see Aldrubi and Unser [1] and the references therein.

When an image possesses *regular scaling* this means that the logarithms of average energies in the detail spaces decay linearly when the resolution of scale increases. The standard 2-D wavelet transform has three detail components, namely horizontal, vertical, and diagonal, and all are characterized by their intrinsic scaling.

In the standard multiresolution hierarchy of images, the representation space A_j is decomposed as

$$A_j = A_{j-1} + H_{j-1} + V_{j-1} + D_{j-1},$$

where the A_{j-1} is the coarser representation and $H_{j-1}, V_{j-1}, D_{j-1}$ are spaces of horizontal, vertical, and diagonal details. This representation is nested, and the coarse representation space A_{j-1} can be further split in the same fashion. Assume that the direction (horizontal, vertical or diagonal) of detail spaces is fixed. Suppose that $d_{j;k_1,k_2}$ is the wavelet coefficient at scale j at the location (k_1, k_2) and that E_j is the average of $d^2_{j;k_1,k_2}$ for all (k_1, k_2), i.e., $E_j = \frac{1}{N} \sum_{(k_1,k_2)} d^2_{j;k_1,k_2}$, where N is total number of coefficients at this particular detail space. By convention, j is a dyadic index corresponding to the base 2 logarithm of the scale. The scale decreases (resolution increases) with increasing indices j. Then,

$$\log E_j = \beta_0 + \beta_1 \times j, \tag{3}$$

with the slope β_1 characterizing regular scaling. The parameters β_0 and β_1 in (3) are estimated from the wavelet decomposition by the least-squares linear regression on pairs $(j, \log E_j)$ for a properly selected range of scale indices j, $j_0 \leq j \leq j_1$.

We utilize the intrinsic scaling in natural images to construct informative detail spaces. The algorithm is detailed below.

Algorithm description. The procedure begins with a three-level wavelet decomposition of the original coarse image. We denote the components of the first level decomposition by A_1, H_1, V_1, and D_1, and the components of level 2 and level 3 decomposition correspondingly.

Data sets were transformed to the wavelet domain utilizing the Daubechies 4 wavelet. To estimate the detail components, we then apply the following regression steps on the components (here illustrated only for horizontal detail components):

$$H_1^2 = \beta_0^{(1)} + \beta_1^{(1)} H_2^2\text{\_e} \quad \text{and} \quad H_2^2 = \beta_0^{(2)} + \beta_1^{(2)} H_3^2\text{\_e},$$

where the components are reshaped into vectors and H_2^2_e and H_3^2_e denote $H_2^{(2)}$ and $H_3^{(2)}$ vectors upsampled using the WII method. As we pointed out, the constant $\beta_1 < 0$ is connected to the global Hurst exponent; it describes intrinsic self-similarity of the image. The intercept β_0 depends on the total energy of the image and does not affect scaling. In our analysis, we actually found that the non-intercept model is appropriate.

Next, $\hat{H}_1$ and $\hat{H}_2$ are estimated as

$$\hat{H}_1 = [\hat{\beta}_0^{(1)} + \hat{\beta}_1^{(1)} H_2^2\text{\_e}]^{1/2} \quad \text{and} \quad \hat{H}_2 = [\hat{\beta}_0^{(2)} + \hat{\beta}_1^{(2)} H_3^2\text{\_e}]^{1/2}.$$

The sign of the generated detail coefficients is assigned to be the same as the sign in the observed coefficients in the degraded image. From the scaling property, the scaling coefficient k equals

$$k = \frac{\log E_{j-1} - \log E_j}{j-(j-1)} = \log \frac{E_{j-1}}{E_j}. \tag{4}$$

We then calculate the scaling coefficient k and use it to estimate the horizontal detail component H_0 as follows

$$k = \log\left(\frac{\bar{\hat{H}}_1^{\,2}}{\bar{\hat{H}}_2^{\,2}\_\mathrm{e}}\right) \Rightarrow k = \log\left(\frac{\bar{H}_0^2}{\bar{\hat{H}}_1^{\,2}\_\mathrm{e}}\right).$$

From this, we approximate H_0 as

$$H_0^2 = e^k \cdot \hat{H}_1^2\_\mathrm{e} \quad \text{i.e.} \quad H_0 = [e^k \cdot \hat{H}_1^2\_\mathrm{e}]^{1/2}.$$

The generated object is “rich” in coefficients; to reduce dimension we apply the threshold, setting small detail coefficients to 0. We choose universal threshold (Donoho and Johnstone [7]) and note that other shrinkage strategies are possible.

The threshold $\lambda = \sqrt{2 \log N} \cdot \sigma$ is applied on the estimated wavelet coefficients. As an estimator of σ, we use the median absolute deviation (MAD) from the median:

$$\hat{\sigma} = \frac{1}{0.6745} \cdot \mathrm{MAD}[d^{(f)}] = 1.4828 \cdot \mathrm{median}[|\tilde{d}^{(f)} - \mathrm{median}(\tilde{d}^{(f)}|],$$

where $\tilde{d}^{(f)}$ is the vector of finest detail coefficients associated to the multiresolution subspaces H, V, and D.

This procedure is repeated for the other two detail components (the vertical and the diagonal). Once we have the estimated detail components, we apply the inverse wavelet transform with our original coarse image as the smooth part and estimated detail components as details, to obtain a higher resolution enhanced image.

4 Results

The data analyzed in our study are from the University of South Florida’s Digital Database for Screening Mammography (DDSM) [10]. (For more information see Section 2). We analyzed 8 malignant and 10 benign cases. Each case consists of four images, however in most cases microcalcifications were only found in one breast, hence we only analyzed the two corresponding images. In addition, some cases had multiple microcalcification sites, which resulted in 40 images we analyzed altogether. First, the ROI detection method described in Section 3.1 was applied to each of the images. ROIs were cropped to smaller images, which contained calcifications of approximately 0.05-1mm in dimension (in terms of pixels: 20 × 20 to 50 × 50 pixels). Our main algorithm explained in 3.3 can be applied to $2^n \times 2^m$ pixels size images, it will here be illustrated on 64×64 images.

Figure 1 shows the result of applying the WII algorithm on an image of a malignant case. We notice a significant improvement from level 0 to level 1 and

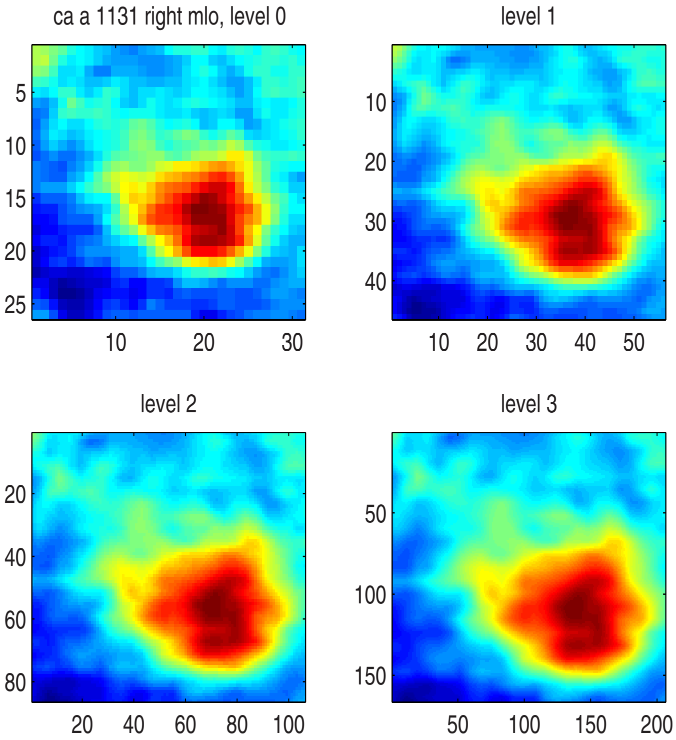


Fig. 1. Results of applying WII method on an image of a malignant calcification

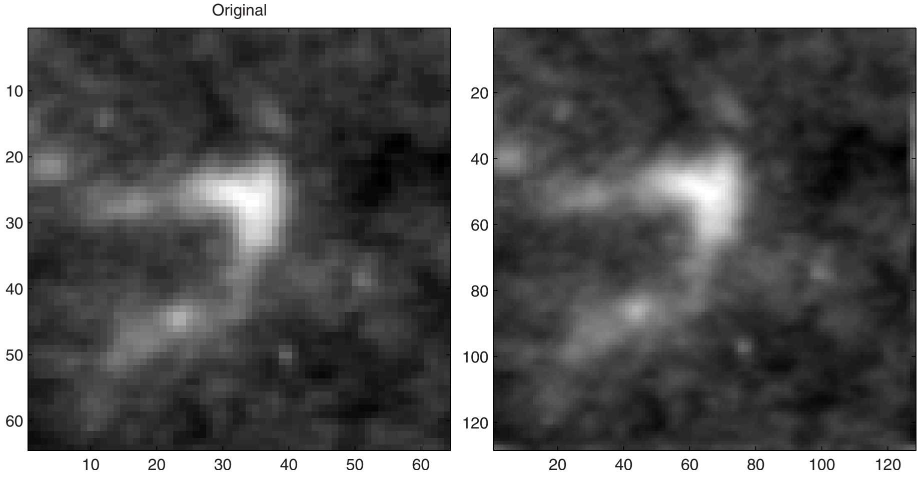


Fig. 2. Results of applying WII algorithm on an image of a malignant calcification(s); left: original image, right: enhanced image

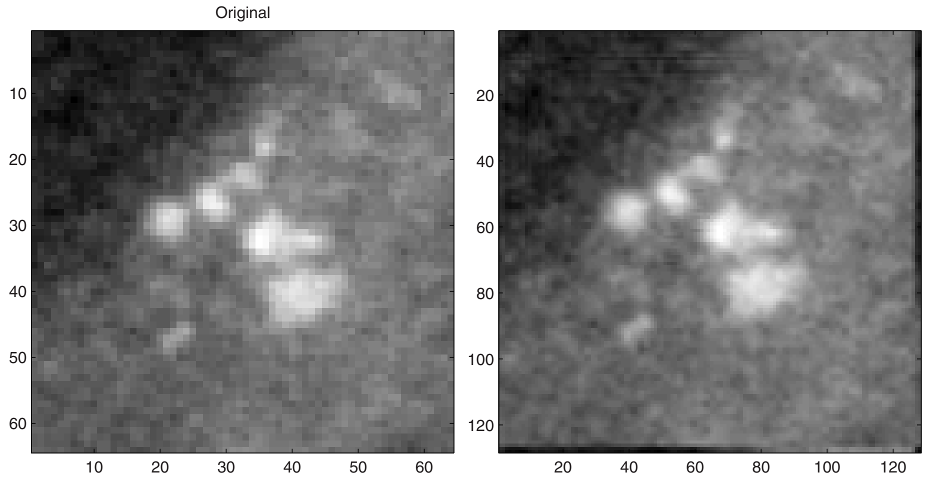


Fig. 3. Results of applying WII algorithm on an image of a benign calcification(s); left: original image, right: enhanced image

from level 1 to level 2, while there is only a very slight difference between levels 2 and 3. The images on Figure 1 were obtained by imputing zeros into the details components of the WII algorithm, with the original image as the smooth part.

Figure 2(left) shows another malignant case image (original image) with a cluster of microcalcifications. Figure 2(right) shows the result of our image enhancement algorithm, described in 3.3, applied on the image in Figure 2(left). We used the gray scale representations of the images since it follows the convention among the radiologists and was preferred by the clinical radiologist on our research team who made the clinical assessments of the images.

Figure 3(left) shows an image of a benign case (original image). Figure 3(right) shows the result of the image enhancement algorithm, applied on the image in Figure 3(left).

5 Conclusion

Motivated by the ubiquitous presence of regular scaling in medical images, we propose a regression-based approach to extrapolate on the wavelet coefficients of an image to be enhanced. The observed image and its extrapolated details are then transformed by an inverse wavelet transform and the image of higher resolution is obtained. This process corresponds to a wavelet-based image interpolation that is improved by information about regular scaling in detail coefficients.

We found that our proposed procedure is efficient and useful in capturing relevant clinical information in the context of digital mammographic imaging. Our assessments were made by evaluations of the 18 knowns preformed by the clinical radiologist on our research team. We emphasize that at this point, the study is more of a feasibility type, than a stringent test of efficacy.

The improved visualization of calcium morphology is expected to translate to greater specificity in assigning degree of suspicion and need-for-biopsy for calcifications noted in mammography. More study with additional observers is planned to confirm what looks to be a novel helpful technology.

We envision several avenues for future algorithm enhancement. We plan to (i) develop new shrinkage strategies to explore alternatives to the traditional universal thresholding, (ii) to identify the wavelet that yields optimal performance for our application, and (iii) to formalize and to expand the evaluation phase with several radiologists involved in a blind repeated measures study design involving an extensive number of cases. We also plan to contrast the WII approach to other approaches based on quantitative or qualitative measures using the same images and the same team of observers.

Acknowledgement. This research was supported by a Georgia Cancer Coalition grant (GCC-141).

References

1. Aldroubi, A. and Unser, M. A. (1996). *Wavelets in Medicine and Biology.*, CRC Press, Boca Raton FL, 616 p.
2. Anastasio, M., Yoshida, H., Nishikawa, R., Giger, M., and Dio, K. (1998). Global Optimization of Wavelet-Based Computer-Aided Diagnosis (CAD) Scheme for the Detection of Clustered Microcalcifications in Digital Mammograms", Kurt Rossmann Laboratories for Radiologic Image Research, Research Report.
3. Bruce, L. M. and Adhami, R. R. (1996). "Classifying Mammographic Mass Shapes Using the Wavelet Transform Modulus-Maxima Method", *IEEE Transactions on Medical Imaging*, pp. 1170–1177, 18 (12).
4. Chang, C-M. and Laine, A. (1996). Coherence of Multiscale Features for Enhancement of Digital Mammograms", *IEEE Transactions on Information Technology in Biomedicine*, 3, 32–46.
5. Coakley, K. and van Doorn T. (1995). Invariant moment shape description of microcalcifications in digital mammograms. *Australas Phys Eng Sci Med.* 18 (2), 114–118.
6. Curpen, B. N., Sickles, E. A., and Sollitto R. A. (1995). The comparative value of mammographic screening for women 40-49 years old versus women 50-59 years old. *AJR*, 164, 1099–1103.
7. Donoho, D. L. and Johnstone, I. M. (1994). "Ideal spatial adaptation via wavelet shrinkage", *Biometrika*, 91:425-455.
8. Egan, R., Sweeney, M., and Sewell, C. (1980). Intramammary calcifications without an associated mass in benign and malignant diseases, *Radiology*, 137, 1–7.
9. Ferrari, R. J. Rangayyan, R. M., Desautels, J. E. L., Borges, R. A. and Frere, A. F. (2004). Automatic Identification of the Pectoral Muscle in Mammograms", *IEEE TRansactions on Medical Imaging*, 23 (2), 232–245.
10. Heath, M., Bowyer, K.W., Kopans, D. *et al.* (1998). Current status of the Digital Database for Screening Mammography, *Digital Mammography*, pp 457–460, Kluwer Academic Publishers.

11. Heath, M., Bowyer, K., Kopans, D., Moore R., and Kegelmeyer P. Jr. (2000). The Digital Database for Screening Mammography, in *The Proceedings of the 5th International Workshop on Digital Mammography* (Toronto, Canada, June 2000), Medical Physics Publishing (Madison, WI), ISBN 1-930524-00-5.
12. Heinlein P., Drexl J., and Schneider W. (2003). Integrated wavelets for enhancement of microcalcifications in digital mammography. *IEEE Trans. Med. Imag.* 22 (3), 402–413.
13. Lado, M. J., Tahoces, P. G., Méndez, P. G., Souto, M. and Vidal, J. J. (1999). A Wavelet-based Algorithm for Detecting Clustered Microcalcifications in Digital Mammograms, *Med. Phys.*, 26 (7), 1294–1305.
14. Lemaur, K., Drouiche, J., and DeConinck, (2003). Highly Regular Wavelets for the Detection of Clustered Microcalcifications in Mammograms, *IEEE Trans. Med. Imag.*, 22(3), 393–401.
15. Mallat, S. (1989). A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Trans. Pattern Anal. Machine Intell.*, 11, 674–693.
16. McLeod G., Parkin G. J. S., and Cowen A. R. (1996). Automatic detection of clustered micro-calcifications using Wavelets, In: , Eds. Doi K, pages 311–316, Publisher: Elsevier Science BV (Amsterdam).
17. Menges, V., Wellauer, J., Engeler, V., and Stadelmann, R. (1973). Correlation of numerically detected microcalcifications on the mammogram and in this manner diagnosed carcinomas and mastopathies, (in German), *Radiologe*. 13 (11), 468–476.
18. Millis, R., Davis, A., Stacey, M., Phil, M. (1976). The detection and significance of calcifications in the breast: A radiologic pathologic study, *Br. J. Radiol.*
19. Seršić, D. and Lončarić, S. (1998). Enhancement of Mammographic Images for Detection of Microcalcifications, *Proceedings of the IX European Signal Processing Conference*, Vol. 2, pp. 693-696, Island of Rhodos, Greece.
20. Smart, C. R., Hendrick, R. E., Rutledge J. H., and Smith, R. A. (1995). Benefit of mammography screening in women ages 40 to 49 years: current evidence from randomized controlled trials, *Cancer*, 75, 1619–1626.
21. Strickland, R. N. and Hahn, H. I. (1996). Wavelet Transform for Detecting Microcalcifications in Mammograms, *IEEE Transactions on Medical Imaging*, Vol. 15, No.2, 218–229.
22. Thangavel, K., Karnan, M., Sivakumar, R. and Mohideen, A. K. (2005). Automatic Detection of Microcalcifications in Mammograms - A Review, *ICGST International Journal on Graphics, Vision and Image Processing, GVIP.*
23. Van de Ville D., Blu T., Unser M. (2005). Isotropic Polyharmonic B-splines: Scaling Functions and Wavelets, *IEEE Transactions on Image Processing*, 14(11), pp. 1798–1813.
24. Wang T. C. and Karayiannis N. B. (1998). Detection of microcalcifications in digital mammograms using wavelets, *IEEE Trans Med Imaging.* 17(4), 498–509.
25. Yoshida, H, Doi, K., Nishikawa, R. M., Ginger, M. L., and Schmidt, R. A. (1996). An Improved Computer-Assisted Scheme Using Wavelet Transform for Detecting Clustered Mirocalcifications in Digital images, *Acad Radiol*, 3, pp. 621–627.

High Level Programming Environment System for Protein Structure Data

Yanchao Wang, Rajshekhar Sunderraman, and Piyaphol Phoungphol

Computer Science Department, Georgia State University, Atlanta, GA 30303, U.S.A
ywang17@student.gsu.edu, raj@cs.gsu.edu,
pphoungphol1@student.gsu.edu

Abstract. In this paper, we present an application system that extends the Object-Oriented Database (OODB) system by adding domain-specific layers to manage protein structure data. Protein-QL, a domain-specific query language, and Protein-OODB layers are added above the OODB. We have implemented this system for protein domain, but we can easily extend it into other biological domains to build a bio-OODBMS. We define protein's primary, secondary, and tertiary structures as internal data types to simplify queries in Protein-QL in such a way that the domain scientists can easily master the query language and formulate data requests. We use EyeDB as the base OODB to communicate with Protein-OODB. Our system uses Java RMI to return results back to the clients so that transactions can be conveniently executed by the clients.

Keywords: Protein Structure Data, Domain Specific Querying, Object-oriented Databases.

1 Introduction

In recent years, protein data have been growing in a very rapid rate due to more and more advanced experimental techniques. As a consequence, storing, organizing, and analyzing huge amounts of protein data in a clear and efficient way is becoming a challenging issue that most protein scientists have to face and solve. Presently, there are a variety of traditional database management systems to manage biological data such as flat files, relational databases, and so on. However, they have limitations.

Flat files usually manage data by using simple text files with rigid formats. The simple tools associated with manipulating these flat files can be easily mastered, but provide little flexibility in dealing with large and complex data sets such as protein structure data. For example, the file has to be processed line by line till the section of the file containing the required data. In addition, flat files do not support complex data types, an important requirement while managing complicated biological data.

Relational databases, on the other hand, are mature and are successfully applied in many areas. One of the major reasons for its success is that the relational data model is much simpler than others. However this advantage becomes a big issue in life science database applications because of the lack of support for complex data types.

Although the developers of these traditional databases design and provide biologists special tools to manage biological data, they need to develop new tools or

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 215–226, 2007.

recode previously developed software often to meet the fast changing data formats and data types. This process requires the biologists to learn new query languages and master new tools and software. Therefore, designing an efficient database management solution for biological data is becoming an urgent task for most biologists and computer scientists.

Since the object-oriented nature of life science data perfectly matches the architecture of object-oriented databases, biologists and computer scientists are switching their attention to object-oriented databases (OODB). Although OODB has above advantage over traditional databases, there are still a lot of problems that need to be solved in order to apply OODB methodologies to manage biological data. One major problem is that the database management systems (DBMS) of OODBs are designed for general-purpose applications and they do not have any built-in data types (ex. protein and nucleic acid) for biological research. They also do not have built-in biological domain-specific functional operations.

In this paper, we present the design and implementation of a system with built-in data types and domain specific functional operators that extend the object-oriented database (OODB) system. Two additional layers, Protein-QL and Protein-OODB are added above an object-oriented database management system (EyeDB). This system is implemented specifically for protein domain, but it is a first step to build a general bio-OODBMS for biological applications. This new system has three components: a client API, a Middleware (including a RMI server, a protein domain specific language Protein-QL and a Protein-OODB), and an object-oriented database management system, EyeDB. Protein-QL provides convenience for protein scientists to store, retrieve, and modify data, and defines some basic operations. Protein-OODB solves some protein data source problems and is used to connect Protein-QL and EyeDB layers. Four internal data types: protein and its three structures, primary, secondary, and tertiary are defined to simplify the queries in Protein-QL. Domain scientists can easily formulate complex requests for data without much learning. Overall, this application system is for a certain biological domain, but it is easy to extend into other biological domains.

The rest of the paper is organized as follows: Section 2 describes the architecture of the high level programming environment system for protein structure data. The details on using the system are described in Section 3. Conclusion and future work are presented in Section 4.

2 High Level Programming System for Protein Structure Data

The architecture of a high level programming environment for protein structure data called Protein-OODBMS is presented in this section. It is a three-layer architecture that consists of the following components: a client API, Middleware (including a RMI server, a query language for protein structures (Protein-QL), and an object-oriented database for protein structures (Protein-OODB)), and an object-oriented database EyeDB layer. Figure 1 illustrates this architecture.

The clients can use this system to send domain specific requests and manage the database. The client writes simple domain specific queries according to Protein-QL syntax and sends them to the server. The server receives the queries and communicates with Protein-QL using RMI and validates query syntax. Then, the query interpreter converts the query into EyeDB OQL queries. Finally, EyeDB sends the results back to the server. This system provides clients convenient access and is easily mastered.

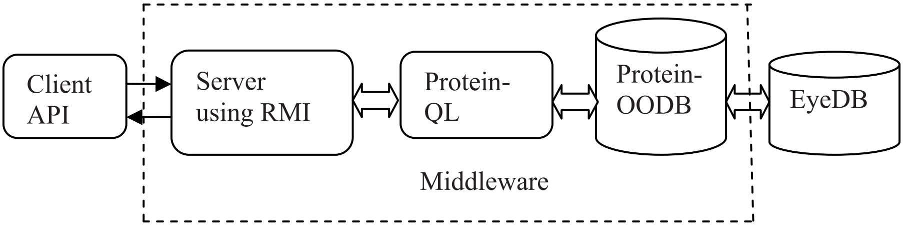
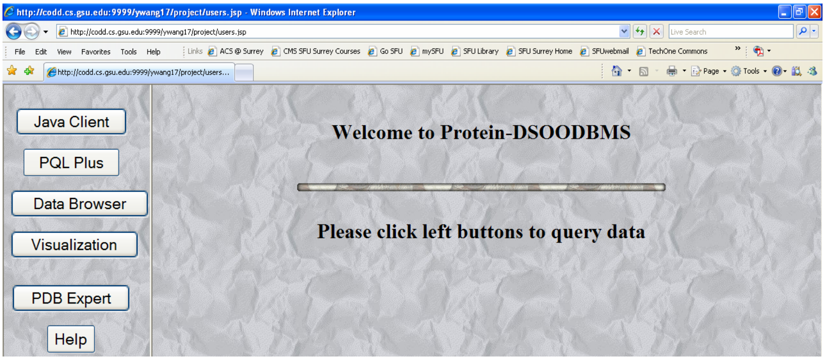


Fig. 1. The architecture of high level programming system for protein structure data

2.1 Client Application

The system provides a variety of client interfaces. Fig. 2 shows the main screen of the client application.


http://codd.cs.gsu.edu:9999/ywang17/project/users.jsp - Windows Internet Explorer

Welcome to Protein-DSOODBMS

Please click left buttons to query data

File Edit View Favorites Tools Help
Links ACS @ Surrey CMS SFU Surrey Courses Go SFU mySFU SFU Library SFU Surrey Home SFUwebmail TechOne Commons
Live Search
Page Tools
Java Client
PQL Plus
Data Browser
Visualization
PDB Expert
Help

Fig. 2. The Client Application

Clients who know Protein-QL syntax can use this system without knowing every detail. The advantages are: (1) Java Client API can easily be viewed and mastered by protein domain scientists who have basic knowledge of Java language without much computer background. Clients are able to formulate Protein-QL queries and have them sent to the server for execution, (2) PQL*Plus Client interaction allows clients to send protein-QL queries directly to Protein-QL without any Java code, (3) Visualization Client is a Protein Explorer tool that allows clients to view protein data structure and functions, and (4) Data Browser provides clients to view protein data in PDB format or object format.

2.2 Middleware Layer

The middleware layer includes a RMI server, a query language for protein structures (Protein-QL), and an object-oriented database for protein structures (Protein-OODB).

Protein-QL

Using Protein-QL, clients can perform basic operations to store, retrieve, and modify protein data and these operations can be executed on basic data types as well as protein data types. Protein-QL defines a list of operators in protein terms, which enables domain scientists to query information in their own language without much syntactical restriction. For example, clients can use the expression `get-Primary(proteinName)` to view the primary structure of protein named "`proteinName`". Our system also provides other operators such as nearest neighbors, cluster of protein, sequences of protein, etc.

The system is implemented based on the features of EyeDB [2] to store complex protein data as objects. EyeDB provides Object Query Language (OQL) to serve the client requests. The Protein-QL interpreter converts Protein-QL queries to OQL queries. Protein-QL not only provides basic operators for EYEDB such as +, -, *, / and % but also supports domain specific operators on protein data types `Primary`, `Secondary`, `Tertiary` and `Protein` that are defined as internal data types. Thus biologists can easily request the information with these domain specific terms. The following are EyeDB definitions for the internal data types:

```
class Protein {string proteinName; string types; string
functions; string remark; set<words *> domainOfPrimaryStru;
Primary * primary; Secondary * secondary; Tertiary * tertiary;
constraint<unique> on proteinName; constraint<notnull> on
proteinName; index on proteinName;};
```

```
struct words {string name;};
```

```
class Primary {string proteinName; array<PrimaryPartition *>
primary;};
```

```
class PrimaryPartition {string name; int serNum; string
chainName; long elementNo; string seq;};
```

```
class Secondary {string proteinName; array<SecondaryPartition *>
secondary;};
```

```
class SecondaryPartition {string name; int serNum; string ID;
int numStrands; string initResName; string initChainID; int
initSeqNum; string endResName; string endChainID;  int
endSeqNum; int sense; string curAtom;  string curResName; string
curChainID; int curResSeq;  string prevAtom; string prevResName;
string prevChainID; int prevResSeq; string comments; int length;
};
```

```
class Tertiary {string proteinName; array<TertiaryPartition *>
tertiary;};
```

```
class TertiaryPartition {string name;  int serNum;  string
atom1; string resName; string chained; int resSeq; double x;
double y; double z; double possib; double pos;string atom2;};
```

We now present some query examples in Protein-QL:

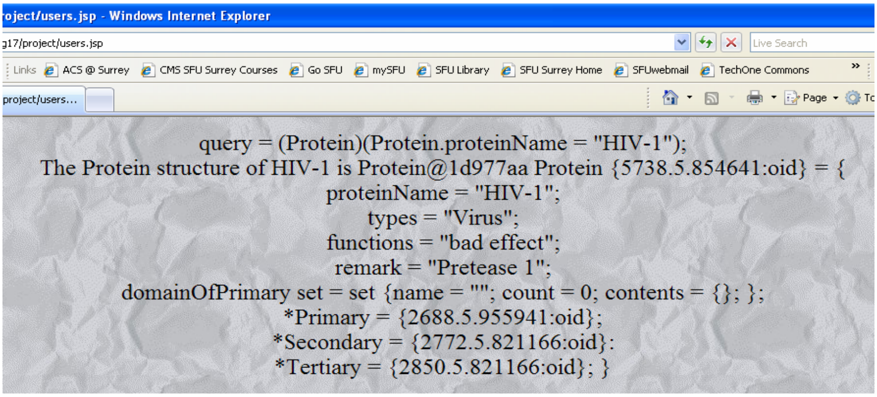
Example 1: Get the protein named “HIV-1” (Abbreviation “HIV-1 Protease”).

```
   (Protein)(Protein.proteinName = “HIV-1”);
```

Then the interpreter generates object query in EyeDB:

```
select p from p in Protein where p.proteinName=“HIV-1”;
```

The result is shown in Fig. 3.



query = (Protein)(Protein.proteinName = "HIV-1");
The Protein structure of HIV-1 is Protein@1d977aa Protein {5738.5.854641:oid} = {
proteinName = "HIV-1";
types = "Virus";
functions = "bad effect";
remark = "Pretease 1";
domainOfPrimary set = set {name = ""; count = 0; contents = {}; };
*Primary = {2688.5.955941:oid};
*Secondary = {2772.5.821166:oid};
*Tertiary = {2850.5.821166:oid}; }

Fig. 3. The result for example 1

The PDB format result may be obtained using the service `PDBFormat (proteinName)`.

Example 2: Get the secondary structure of protein named “HIV-1”.

```
   (Protein.secondary)(Protein.proteinName= “HIV-1”);
```

Our interpreter will generate object query in EyeDB:

```
select p.secondary from p in Protein where p.proteinName
= “HIV-1”;
```

The result is shown in the Fig. 4.

The result may also be viewed in PDB format using `PDBSecondary` service.

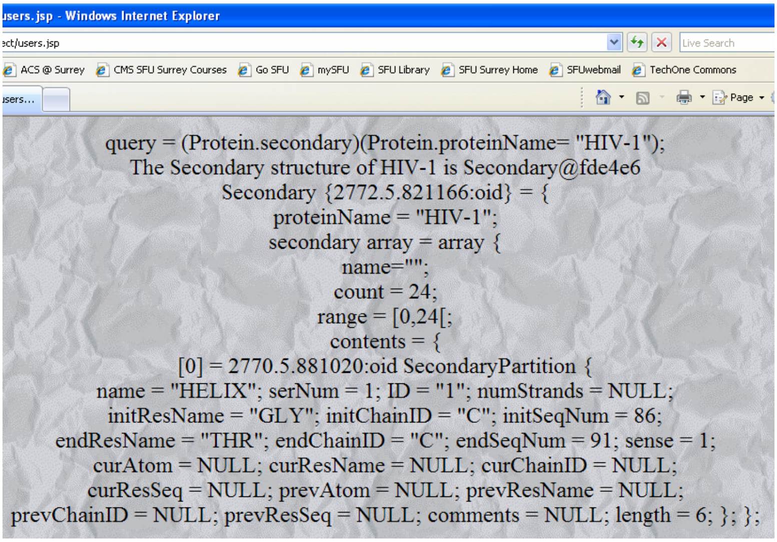
Example 3: Delete a protein object whose name is “HIV-1”

```
   delete (Protein)(Protein.proteinName = “HIV-1”);
```

The interpreter generates object query in EyeDB:

```
select Protein.proteinName="HIV-1";
=5738.5.854641:oid
delete 5738.5.854641:oid;
= 5738.5.854641:oid
```

The result is: `= 5738.5.854641:oid`.



```
query = (Protein.secondary)(Protein.proteinName= "HIV-1");
The Secondary structure of HIV-1 is Secondary@fde4e6
Secondary {2772.5.821166:oid} = {
proteinName = "HIV-1";
secondary array = array {
name="";
count = 24;
range = [0,24[;
contents = {
[0] = 2770.5.881020:oid SecondaryPartition {
name = "HELIX"; serNum = 1; ID = "1"; numStrands = NULL;
initResName = "GLY"; initChainID = "C"; initSeqNum = 86;
endResName = "THR"; endChainID = "C"; endSeqNum = 91; sense = 1;
curAtom = NULL; curResName = NULL; curChainID = NULL;
curResSeq = NULL; prevAtom = NULL; prevResName = NULL;
prevChainID = NULL; prevResSeq = NULL; comments = NULL; length = 6; }; };
```

Fig. 4. The result for example 2

Protein-OODB

With advancement in the development of the new laboratory instruments and experimental techniques we have seen an explosion in protein data. These protein data may come from different computational techniques, experiments, and interpretation of primary data and the data sources themselves may include a variety of different information that may result in heterogeneous protein data sources. Therefore, our system provides the following ways to solve these heterogeneity problems: (a) We use the most confident data sources by identifying the confidence of data sources to solve multiple sources problems, (b) We provide a data dictionary [11] (sometimes called a controlled vocabulary) to list synonyms, homonyms, common misspellings, and abbreviations such that system can search them quickly in Table 1. Our data dictionary defines some domain specific concepts, terms, and interpretation, which

Table 1. Data Dictionary

Synonym	Homonym	Common Misspelling	Abbreviation	Same Object with Different Name
HIV-1/ Human immunodeficiency virus 1/ Human Immunodeficiency Virus Type 1/ HIV-I/ Immunodeficiency Virus Type 1, Human		H1V-1	HIV-1	
……	……	……	……	……

Table 2. The operations and interpretation of some query operators in Protein-QL

Operator	**Interpretation**
sequence(String pn) //pn is ProteinName	sequence: {s l s is the sequence of protein pn's primary structure—amino acid sequence}
oneLetterCode(String pn)	oneLetterCode: {olc l olc is 1-letter code of protein pn's primary structure}
lengthOfSequence(String pn)	lengthOfSequence: {l l l is the length of protein pn's primary structure}
subSequence(String pn, int start, int end)	subSequence: {sub l sub is the subsequence of pn from position *start* to position *end* in p's primary structure}
location (String pn, string sub)	location: {pos l sub is the subsequence in three letter of pn, pos is the first location of *sub* occurring in pn's primary structure}
locationOneLetter(String pn, String sub)	locationOneLetter : {pos l sub is the subsequence in one letter of pn, pos is the first location of *sub* occurring in pn's primary structure}
noOfSub(String pn, String sub)	noOfSub : {n l *sub* is the subsequence of pn, n is the number of sub in pn's primary structure}
noOfSubOneLetter(String pn, String sub)	noOfSubOneLetter : {n l *sub* is the subsequence in one letter of pn, n is the number of sub in pn's primary structure}
globalAlignment(String pn1, String pn2)	globalAlignment: {ga l ga is the global alignment using Needleman-Wunsch method in pn1, pn2's primary structure}
mostSimilarity(String pn)	mostSimilarity: {ms l ms is the most similar protein of pn according to Needleman-Wunsch global alignment}
getPrimary (String pn)	getPrimary: {ps l ps is pn's primary structure}
noOfHelix(String pn)	noOfHelix: {n l n is the number of helix in pn's secondary structure}
lengthOfHelix(String pn, int helix)	lengthOfHelix: {l l l is the length of *helix* of protein pn's secondary structure}
lengthOfEachHelix(String pn)	lengthOfEachHelix: {l(List) l l are the length of all *helix* of protein pn's secondary structure}
noOfChain(String pn)	noOfChain: {n l n is the number of chains in pn's secondary structure}
noOfStrand(String pn, String chain)	noOfStrand: {n l n is the number of strands of *chain* in pn's secondary structure}
noOfStrandEachChain(String pn)	noOfStrandEachChain: {n(List) l n are the number of strands of all *chains* in pn's secondary structure}
senseOfEachStrand(String pn)	senseOfEachStrand: {sense(List) l sense are the senses of all *strands* with respect to previous strand in the sheet in pn's secondary structure}
getSecondary(String pn)	getSecondary: {ps l ps is pn's secondary structure}
center3D(String pn)	center3D: {d l d is the center coordinate of pn}
nearestNeighbor3D(String pn)	nearestNeighbor3D: {p1 l p1 has the shortest distance among neighbors of pn in tertiary structure}
cluster(String pn,double d)	cluster: {p1(List) l the distance between pn in tertiary structure and p1 is equal to or less than d}
getTertiary(String pn)	getTertiray: {ps l ps is pn's tertiary structure}
getProtein(String pn)	getProtein: {protein l protein is pn's all information that is stored in databases}

make databases easier and more efficient to search without any guessing work. It also can solve the problem that the same object has different names in life sciences, and (c) We develop a schedule table to store the queries needing very long time to execute. These queries and their results are updated with latest data to keep them up-to-date. Clients can get them from local memory to save time. We define data dictionary as follows:

```
class DataDictionary{string proteinName;
set<words *> synonym;  set<words *> homonyms;
set<words *> cmp; //Common Misspelling
set<words *> abbreviation;
set<words *> sodn;  //Same Object with Different Name
};
```

2.3 Protein Structure Data Operators

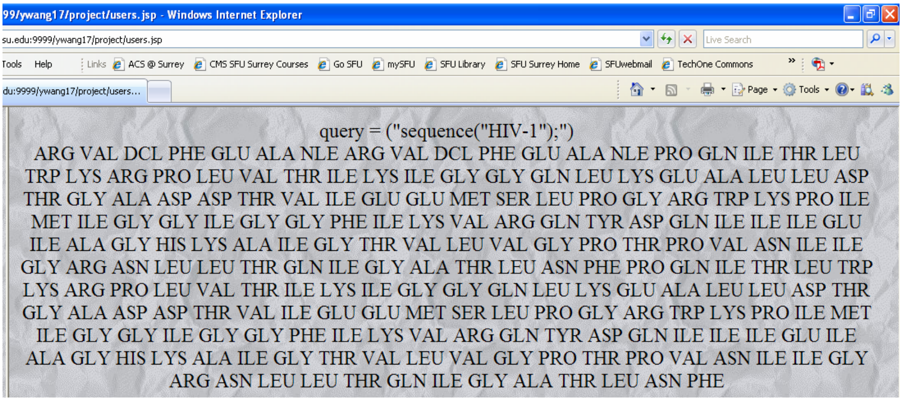
As we know, protein data is very large and clients often need only parts of the data. So, our protein-OODB layer provides operators based on the protein domain, which clients can conveniently and easily use in the queries without any extra learning. Some operators are shown in the Table 2; Users can easily add more operators and functions in the future.

These operators can be used in Protein-QL as the following examples illustrate.

Example 4: Get the sequence of protein named "HIV-1".

```
sequence("HIV-1");
```

The result is shown in following Fig. 5.



```
query = ("sequence("HIV-1");")
ARG VAL DCL PHE GLU ALA NLE ARG VAL DCL PHE GLU ALA NLE PRO GLN ILE THR LEU
TRP LYS ARG PRO LEU VAL THR ILE LYS ILE GLY GLY GLN LEU LYS GLU ALA LEU LEU ASP
THR GLY ALA ASP ASP THR VAL ILE GLU GLU MET SER LEU PRO GLY ARG TRP LYS PRO ILE
MET ILE GLY GLY ILE GLY GLY PHE ILE LYS VAL ARG GLN TYR ASP GLN ILE ILE ILE GLU
ILE ALA GLY HIS LYS ALA ILE GLY THR VAL LEU VAL GLY PRO THR PRO VAL ASN ILE ILE
GLY ARG ASN LEU LEU THR GLN ILE GLY ALA THR LEU ASN PHE PRO GLN ILE THR LEU TRP
LYS ARG PRO LEU VAL THR ILE LYS ILE GLY GLY GLN LEU LYS GLU ALA LEU LEU ASP THR
GLY ALA ASP ASP THR VAL ILE GLU GLU MET SER LEU PRO GLY ARG TRP LYS PRO ILE MET
ILE GLY GLY ILE GLY GLY PHE ILE LYS VAL ARG GLN TYR ASP GLN ILE ILE ILE GLU ILE
ALA GLY HIS LYS ALA ILE GLY THR VAL LEU VAL GLY PRO THR PRO VAL ASN ILE ILE GLY
ARG ASN LEU LEU THR GLN ILE GLY ALA THR LEU ASN PHE
```

Fig. 5. The result for `sequence("HIV-1")`

Example 5: Get the number of helix of protein named "HIV-1".

```
noOfHelix("HIV-1");
```

The result: *2*.

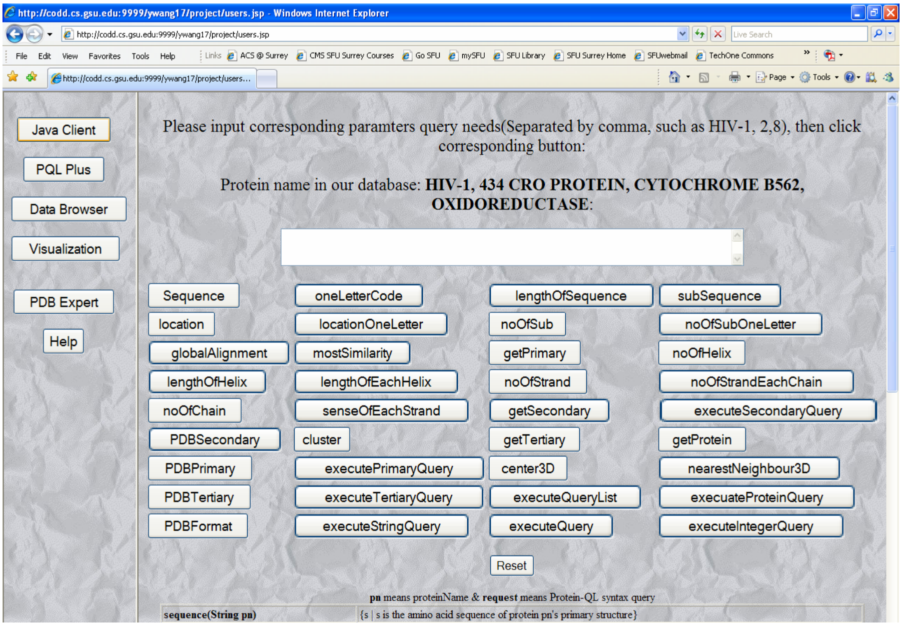
Example 6: Get the nearest neighbor of protein named “HIV-1”.

```
nearestNeighbor3D(“HIV-1”);
```

The result: `Protein{5738.5.854641:oid }={proteinName=……}`

3 Details on Using System

The Java clients only need to know Protein-QL syntax, service name, input parameters and output. The details of implementation do not affect the usage of clients. In the implementation, the system developed algorithm to create an alignment from a number of sequences such that the system does not need to extract the data for each protein from the database to increase performance (because less data has to be read from and written to the database) and reduce storage needs. We also designed algorithm to calculate center of protein and store them in the file, therefore, the system only needs to calculate center one time for each protein. The Java Client interface is shown in Fig. 6.



Please input corresponding paramters query needs(Separated by comma, such as HIV-1, 2,8), then click corresponding button:

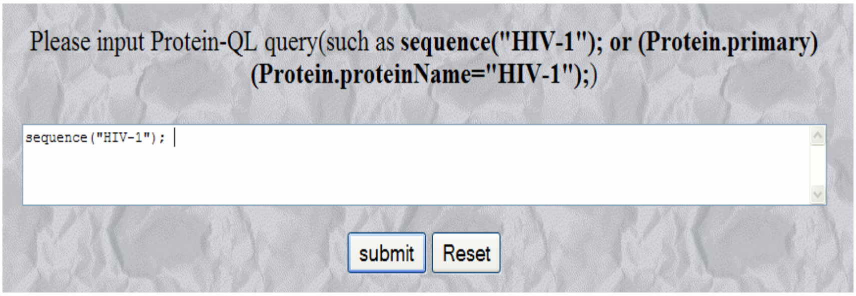
Protein name in our database: **HIV-1, 434 CRO PROTEIN, CYTOCHROME B562, OXIDOREDUCTASE**:

pn means proteinName & **request** means Protein-QL syntax query

sequence(String pn)	{s | s is the amino acid sequence of protein pn's primary structure}

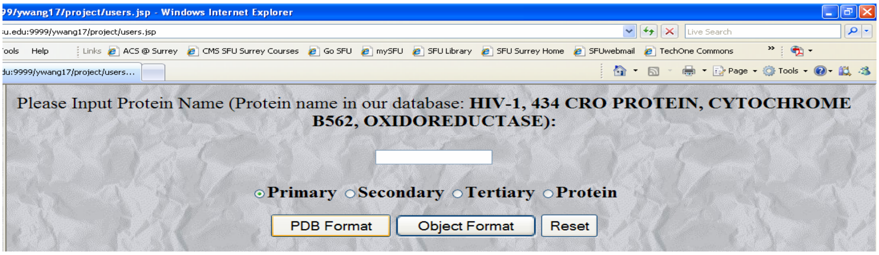
Fig. 6. Java Client Interface

Fig. 7 shows the Protein-QL*Plus interface. Protein-QL*Plus can use above services to get results from our database. Clients only need to input Protein-QL queries according to Protein-QL syntax.


Please input Protein-QL query(such as **sequence("HIV-1"); or (Protein.primary)(Protein.proteinName="HIV-1");)**

```
sequence("HIV-1");
```

Fig. 7. Protein-QL Plus Interface

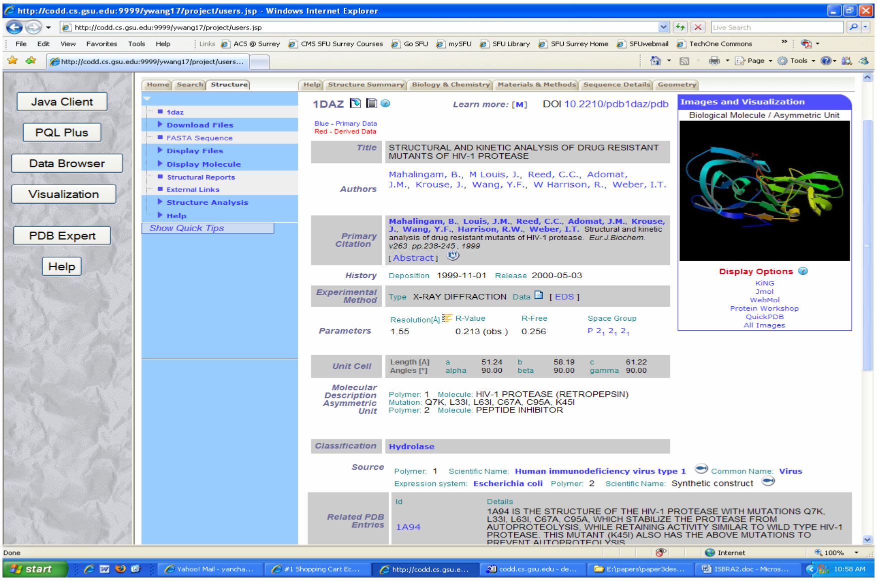

99/ywang17/project/users.jsp - Windows Internet Explorer
u.edu:9999/ywang17/project/users.jsp
Live Search
Tools Help Links ACS @ Surrey CMS SFU Surrey Courses Go SFU mySFU SFU Library SFU Surrey Home SFUwebmail TechOne Commons
du:9999/ywang17/project/users...
Page Tools

Please Input Protein Name (Protein name in our database: **HIV-1, 434 CRO PROTEIN, CYTOCHROME B562, OXIDOREDUCTASE**):

● **Primary** ○ **Secondary** ○ **Tertiary** ○ **Protein**

PDB Format
Object Format
Reset

Fig. 8. Data Browser API



1DAZ Learn more: [M] DOI 10.2210/pdb1daz/pdb

Blue - Primary Data
Red - Derived Data

Title	STRUCTURAL AND KINETIC ANALYSIS OF DRUG RESISTANT MUTANTS OF HIV-1 PROTEASE
Authors	Mahalingam, B., M Louis, J., Reed, C.C., Adomat, J.M., Krouse, J., Wang, Y.F., W Harrison, R., Weber, I.T.
Primary Citation	**Mahalingam, B., Louis, J.M., Reed, C.C., Adomat, J.M., Krouse, J., Wang, Y.F., Harrison, R.W., Weber, I.T.** Structural and kinetic analysis of drug resistant mutants of HIV-1 protease. *Eur.J.Biochem. v263 pp.238-245 , 1999* [Abstract]
History	Deposition 1999-11-01 Release 2000-05-03
Experimental Method	Type X-RAY DIFFRACTION Data [EDS]
Parameters	Resolution[Å] 1.55 R-Value 0.213 (obs.) R-Free 0.256 Space Group P 2_1 2_1 2_1
Unit Cell	Length [Å] a 51.24 b 58.19 c 61.22 Angles [°] alpha 90.00 beta 90.00 gamma 90.00
Molecular Description Asymmetric Unit	Polymer: 1 Molecule: HIV-1 PROTEASE (RETROPEPSIN) Mutation: Q7K, L33I, L63I, C67A, C95A, K45I Polymer: 2 Molecule: PEPTIDE INHIBITOR
Classification	**Hydrolase**
Source	Polymer: 1 Scientific Name: **Human immunodeficiency virus type 1** Common Name: **Virus** Expression system: **Escherichia coli** Polymer: 2 Scientific Name: Synthetic construct

Related PDB Entries	Id	Details
	1A94	1A94 IS THE STRUCTURE OF THE HIV-1 PROTEASE WITH MUTATIONS Q7K, L33I, L63I, C67A, C95A, WHICH STABILIZE THE PROTEASE FROM AUTOPROTEOLYSIS, WHILE RETAINING ACTIVITY SIMILAR TO WILD TYPE HIV-1 PROTEASE. THIS MUTANT (K45I) ALSO HAS THE ABOVE MUTATIONS TO PREVENT AUTOPROTEOLYSIS

Fig. 9. Visualization Interface connected to RCSB PDB

We also provide a data browser interface that allows domain scientist to easily view all aspects of the protein structure data. The data browser interface is shown in Fig. 8.

Finally, we also provide a visualization interface that can link users to PDB 3D view of protein. The interface is shown in Fig. 9.

4 Conclusions and Future Work

This paper describes an application system that is protein domain specific and implemented by adding some new features (such as internal protein data types) to existing OODB--EyeDB. This system has three components: a client interface, a middleware (including a server, Protein-QL, and a Protein-OODB), and an EyeDB database backend. Protein-QL provides convenience for users to store, retrieve, and modify data. The system defines four protein internal data types to simplify the queries so that the users can easily understand and send requests. The system also defines some basic and domain specific operations. Protein-OODB solves some protein data source problems and is used to connect Protein-QL and EyeDB layers. This application paper presents a general idea to develop a new system for a certain biological domain. It is very easy to extend this system into other domains.

In the future, we propose to add data curation module ([9]) to enable domain scientists to detect and fix errors and inconsistencies in the protein structure data. The data curation system is tied to a data input system that takes protein structure data in PDB format as input and through a series of semi-automated steps converts the data into object-oriented data to be store din EyeDB. In addition, we propose to extend the system to include data from other related domain such as genome data.

References

1. Yanchao Wang, Rajshekhar Sunderraman and Hao Tian. A Domain Specific Data Management Architecture for Protein Structure Data. The 28th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2006.
2. E. Viara, E. Barillot, and G. Vaysseix. The EyeDB OODBMS, International Database Engineering and Applications Symposium, Montreal, IEEE Publications, Pages 390-402.
3. Dan E. Krane, and Michael L. Raymer. Fundamental Concepts of Bioinformatics. 2003 Pearson Education, Inc., publishing as Benjamin Cummings.
4. David W. Mount, Bioinformatics. Sequence and Genome Analysis. 2001 by Cold Spring Harbor Laboratory Press, Cold Spring Harbor, New York.
5. C. Branden. Introduction to Protein Structure. Published by Garden Publishing, Inc.
6. Ingvar Eidhammer, Inge Jonassen, William R. Taylor. Protein Bioinformatics: An Algorithm Approach to Sequence and Structure Analysis. Pub by John Wiley & Sons.
7. R. Esser, J.W. Janneck. A Framework for Defining Domain-Specific Visual Languages. In Workshop on Domain Specific Visual Languages, in conjunction with ACM Conference on OOPSLA-2001.

8. E. F. Codd. A relational model of data for large shared data banks. Commun. ACM (1970), Vol. 13, page 377–387.
9. Yanchao Wang, Rajshekhar Sunderraman. PDB Data Curation. The 28th Annual International Conference of the IEEE EMBS, 2006.
10. http://www.whatislife.com/reader/protein/protein.html
11. http://www.boxesandarrows.com/archives/what_is_a_controlled_vocabulary.php

Finding Minimal Sets of Informative Genes in Microarray Data

Kung-Hua Chang, Yong Kyun Kwon, and D. Stott Parker

Department of Computer Science, University of California, Los Angeles,
Los Angeles, CA 90095-1596, USA
{kunghua,kwon624,stott}@cs.ucla.edu

Abstract. For a microarray dataset with attached phenotype information – which gives expression levels of various genes and a phenotype classification for each of a set of samples – an important problem is to find *informative genes*. These genes have high information content as attributes for classification, minimizing the expected number of tests needed to identify a phenotype. This study investigates the use of a heuristic method for finding complete sets of informative genes (sets that are sufficient for constructing a maximally discriminating classifier) that are as small as possible. These *minimal sets of informative genes* can be very useful in developing an appreciation for the data. Our method uses branch-and-bound depth-first search. Experimental results suggest that our method is effective in finding minimal gene sets, and the resulting classifiers have good performance in terms of classification accuracy.

1 Introduction

Microarray analysis usually involves thousands or tens of thousands of genes. An important goal is to select informative genes that can subsequently be used as biomarkers. An example would be cancer microarray datasets, where the extracted biomarkers could be used in detecting cancer or finding the root cause of cancer. Such a dataset can be viewed as table whose columns represent genes, and whose rows represent samples with an associated phenotype classification (cancer status).

A natural problem in this area is to find genes that are informative – i.e., useful in classification, providing lots of information toward identifying the phenotype class. Many methods have been used to extract informative genes from microarray datasets. One kind of method is a greedy method that uses some statistical evaluation of each gene, such as *t*-test [3], signal-to-noise ratio [6], or SAM [11]. An advantage of this kind of method is that the computational cost is low. A drawback is that greedy methods always select redundant genes because they only evaluate individual genes, without considering correlations between genes.

In this paper we consider the problem of finding minimal sets of informative genes that are complete, i.e., which can be used in constructing a classifier for the input data. Since searching all possible sets of genes is generally infeasible, a heuristic strategy can be used to partially search combinations of genes and evaluate their performance.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 227–236, 2007.

One example of heuristic search methods is genetic algorithms [8]. A genetic algorithm starts with an initial set of "genetic parameters" defining a search configuration (candidate solution) that are used in the process of mutation and crossover in order to produce new sets of parameters that pass a pre-defined fitness function. The algorithm terminates when there is no improvement over several generations [8]. Advantages of heuristic search methods include that they generally do not select many redundant genes, and this could lead to finding the optimal gene subset. Drawbacks include that they are sensitive to noise, prone to overfitting, and computationally expensive.

We next describe a heuristic search method that uses branch-and-bound depth-first search approach to extract minimal sets of informative genes from microarray data. The general idea of our algorithm is to avoid selecting redundant genes. We perform experiments on 7 publicly available cancer microarray datasets and compare our results with results in [8].

2 Heuristic Branch-and-Bound Depth First Search

Suppose we have a classification learning problem $L = \{(X_\mathrm{i}, Y_\mathrm{i}) \mid i = 1, 2, \ldots, N\}$ that has N training instances. X_i is the vector of attributes for the i-th training instance, while $Y_\mathrm{i} \in \{1, 2, \ldots, C\}$ is the class label for the i-th training instance.

Definition 1. Suppose X_ij is the j-th attribute of the i-th training instance, so that we can write $X_i = \{X_{ij} \mid j = 1, 2, \ldots, M\}$. The **distribution of the training instance** for the j-th attribute can be computed as in (1):

$$D_{ij} = 1 \text{ if class value } Y_i \text{ is discriminated by attribute } X_j\text{, 0 otherwise.} \tag{1}$$

That is, D_j is a bit vector of 0s and 1s that contains the classification ability of the j-th attribute for the N training instances. Thus the i-th entry in D_j is 1 if the j-th attribute can correctly discriminate the class label Y_i from other class labels, and 0 if it cannot.

Definition 2. Define $ACC(D_j)$ as the **classification accuracy** of the j-th attribute, giving the number of 1s in D_j. If necessary we reorder the column vectors D_j in D in decreasing order of $ACC(D_j)$, so that attributes with better classification accuracy on the N training instances appear first in D.

Definition 3. Set D_{max} as the element-wise maximum (equivalently: least upper bound, element-wise "or") of the D_j, $j = 1, 2, \ldots, M$ – so that D_{max} is a bit vector that represents the maximal classification ability of any deterministic classifier obtained by combining the individual classification abilities of all attributes, as in (2):

$$D_{max} = \text{“}UNION\text{”}(D) = D_1 \cup D_2 \cup \cdots \cup D_M \tag{2}$$

Our algorithm starts with an empty set Q. It then considers each D_j (in descending order of classification accuracy) as the root of a search tree, includes D_j in Q, and performs the depth-first search in Algorithm 2. This search then starts by recursively choosing the D_k that most increases classification accuracy, as in (3):

$$\text{choose } D_k \text{ where } k \in argmax_j \,(\, \mathrm{ACC}(\, Q \cup \{ D_j \}) \,) \quad (\, j = 1, 2, \ldots, M\,). \tag{3}$$

Thus, D_k is chosen so as to eliminate as many 0s in Q as possible, providing the greatest improvement in classification accuracy. The algorithm then includes D_k in Q, and continues recursively. Whenever "UNION"$(Q) = D_{max}$, signaling the end of search, the algorithm records the classification accuracy $ACC(Q)$ on the training and test sets and then backtracks.

```
Algorithm 1   root_node_selection :   Select Root Node From D After Sort
sort the vectors $D_j$ in decreasing order of $ACC(D_j)$
$\pi = ACC(D_{max}) - ACC(MAXACC(D))$
$\varepsilon = 1/M \sum_j ACC(D_j)$
for $j$ = 1 to $M$ do
    $Q \leftarrow \{ D_j \}$
    $BBDFS( Q, \pi, \varepsilon )$

Algorithm 2   BBDFS(Q, π, ε):   Branch-and-Bound Depth-First Search
if total number of solutions > $T$:
    Stop
if "UNION"$(Q) = D_{max}$ :
    Train classifier using the attributes whose vectors appear in the set $Q$
    Classify training and test set with this classifier and compute $ACC(Q)$
    if $ACC(Q) < \varepsilon$:
        terminate BBDFS    [and return to root_node_selection]
    else return

$Qc \leftarrow \{ D_k \mid k \in argmax_j ACC(Q \cup \{ D_j \}) \}$
if $ACC(Q \cup \{ MAXACC(Qc) \}) - ACC(Q) < \pi$ :
    $L = n$
else $L$ = number of vectors in $Qc$
for each of the first $L$ vectors $D_j$ in $Qc$ do
    $BBDFS(Q \cup D_j, \pi, \varepsilon)$
```

Fig. 1. Algorithms 1 and 2

The algorithm may not terminate, even with our greedy heuristic. So, in order to improve performance we use a branch-and-bound method to prune nodes and branches, and also limit the total number of solutions to a given parameter T.

Definition 4. Define a branching threshold $\pi = ACC(D_{max}) - ACC(MAXACC(D))$ where $MAXACC(D)$ is the vector D_j that has maximum $ACC(D_j)$ value.

During expansion of D_k, if $ACC(Q \cup \{ MAXACC(Qc) \}) - ACC(Q) < \pi$, we limit the expansion of nodes to the first n nodes, where Qc is the queue of promising attributes

D_j computed in Algorithm 2 and n is another parameter. In other words, if the increase in accuracy gained from adding attribute k is less than the predefined branching threshold π, we only expand the first n nodes. Since we sort the vectors in D in decreasing order of $ACC(D_j)$, the first n nodes (attributes) have better classification ability.

Definition 5. Define a pruning threshold ε as the average classification ability of each attribute as in (4):

$$\varepsilon = \frac{1}{M}\sum_{j}^{M} ACC(D_j) \tag{4}$$

If $Q = D_{max}$ and $ACC(Q) < \varepsilon$, then we abandon the current expansion and jump to the next root node selection process. That is, if the classification accuracy of the current attribute subset is less than the average classification accuracy of each individual attribute, we abandon the whole attribute subset and start from the next root node expansion. For more detail please refer to Algorithms 1 and 2 in Figure 1.

3 Experimental Setup and Results

3.1 Preprocessing Steps

We use the same preprocessing steps as in [8]. Since the downloaded raw microarray data may have abnormal values, we apply the minimum (θ_{min}) and maximum (θ_{max}) values as thresholds. If the value is less than θ_{min}, we replace the value with θ_{min}. If the value is greater than θ_{max}, we replace the value with θ_{max}. If there are missing values present, we use the k-nearest-neighbor method to fill in these missing values. We then use a variation filter to exclude genes g that violate max(g) – min(g) > Δ and max(g)/min(g) > Ω, where max(g) and min(g) are the maximum and minimum gene expression values of g across samples, and θ_{min},θ_{max}, Δ and Ω are parameters that depend on the input data. In the experiments, we linearly scaled all expression values in the range [0, 1], calculating the scaled value g' as:

$$g' = (g - \min(g)) / (\max(g) - \min(g)). \tag{5}$$

The classifier we chose was the support vector machine (SVM), and in particular chose the implementation of LIBSVM [5]. The kernel we used was the linear kernel with default parameters. The only variations we considered were between use of C-support vector classification (C-SVC) and ν-support vector classification (ν-SVC). We used ν-SVC for the Brain Cancer, Prostate Cancer, Central Nervous System, and Colon Cancer datasets, and used C-SVC for the Lung Carcinoma, ALL-MLL Leukemia, and MLL Leukemia datasets (described below). The parameters for our BBDFS algorithm were $T = 100000$, $n = 30$. That is, we only chose the first 100000 attribute subsets, and expanded only the first 30 nodes in the queue Qc whenever the branching threshold π was not satisfied.

3.2 Datasets

Lung Carcinoma. The lung carcinoma data set [4] has 12600 genes in 203 samples. The 203 samples consist of 139 lung adenocarcinomas (AD), 21 squamous (SQ) cell carcinoma cases, 20 pulmonary carcinoid (COID) tumors and 6 small cell lung cancers (SCLC), as well as 17 normal lung (NL) samples. Negative gene expressions are replaced by zero. Our variation filter used a standard deviation threshold of 50 expression units, and 3312 genes were selected from 12600 genes. Then we rescaled the data and divided it randomly into mutually exclusive training sets consisting of 102 samples and test sets of 101 samples [8]. This dataset is a 5-class classification problem.

Brain Cancer. The brain cancer data set [7] contains 12625 genes and 50 gliomas samples: 28 glioblastomas and 22 anaplastic oligodendrogliomas divided into two subsets of classic and non-classic gliomas. The classic subset contains 14 glioblastomas and 7 anaplastic oligodendrogliomas with classic histology, and it is used as a training set to predict the classes of clinically common, histologically non-classic samples consisting of 14 glioblastomas and 15 anaplastic oligodendrogliomas samples [8]. After preprocessing withθ_{min} = 20, θ_{max} = 16000, Δ = 100 and Ω = 3, only 4434 genes were left. Then we linearly scaled the values [8]. This dataset is a 2-class classification problem**.**

Prostate Cancer. The prostate cancer dataset [10] contains gene expressions profiles derived from 52 prostate tumors and 50 non-tumor prostate (normal) samples with approximately 12,600 genes and ESTs. Raw expression values were preprocessed with θ_{min} = 10, θ_{max} = 16000, Δ = 50 and Ω = 5. After preprocessing, only 5966 genes were left, and these were then normalized. Due to unavailability of the independent data set, we divided the initial set into mutually exclusive training and test sets, each containing 50% of the total samples [8]. This dataset is a 2-class classification problem.

Central Nervous System (CNS). The central nervous system dataset [9] contains 7129 genes with 60 patients; 21 out of the 60 are survivors, while 39 out of 60 patients are not. Raw expression values were preprocessed with θ_{min} = 20, θ_{max} = 16000, Δ = 500 and Ω = 5. After preprocessing, 4739 genes are left and then normalized. Since there is no independent test set, we divided the initial set into mutually exclusive training and test set, containing 50% of the total samples. This is a 2-class classification problem.

Colon Tumor. The colon tumor dataset [1] contains 2000 genes selected from 6500 genes with 62 samples from colon-cancer patients; 40 out of 62 tumor biopsies are from tumors, and 20 out of 62 biopsies are from healthy parts of the colon of the same patient. Since the raw expression values have been preprocessed by [1], we kept all 2000 genes and then normalized them. Since there is no independent test set, we divided the initial set into mutually exclusive training and testing sets, each containing 50% of the total samples. This dataset is a 2-class classification problem.

ALL-AML Leukemia. The ALL-AML leukemia dataset [6] contains 7129 genes. The training set contains 38 bone marrow samples with 27 ALL and 11 AML, while the independent test set contains 34 bone marrow samples with 20 ALL and 14 AML. Raw expression values are preprocessed with $\theta_{min} = 1$, $\theta_{max} = 16000$, $\Delta = 500$ and $\Omega = 5$. After preprocessing, 3927 genes were left, and these were then normalized. This is a 2-class classification problem.

MLL Leukemia. The MLL leukemia dataset [2] contains 12582 genes. The training set contains 57 Leukemia samples with 20 ALL, 17 MLL, and 20 AML, while the independent test set contains 15 Leukemia samples with 4 ALL, 3 MLL, and 8 AML. Raw expression values are preprocessed with $\theta_{min} = 100$, $\theta_{max} = 16000$, $\Delta = 500$ and $\Omega = 5$. After preprocessing, 8685 genes were left, and these were then normalized. This dataset is a 3-class problem.

3.3 Experimental Results

Table 1 gives baseline results by using LIBSVM on the training set with all genes and classifying the test set from the above datasets. Table 2 gives the results returned by using LIBSVM with the gene subsets returned by BBDFS algorithm. Detailed experimental results are available at **http://www.cs.ucla.edu/~kunghua/ISBRA2007/**.

Table 1. Classification accuracy using all genes on 7 microarray datasets

Data Set	Training accuracy	Test accuracy	# Genes
Lung Carcinoma	100.00%	93.07%	3312
Brain Cancer	100.00%	62.07%	4434
Prostate Cancer	96.08%	78.43%	5966
CNS	100.00%	62.96%	4739
Colon Cancer	100.00%	70.97%	2000
ALL-MLL	100.00%	88.24%	3927
MLL	100.00%	100.00%	8685

By comparing tables 1 and 2, we see that the BBDFS algorithm is superior not only in finding very small gene subsets, but also in improving classification accuracy. The only exception is the MLL Leukemia dataset, but the results are very close. These results also suggest that many genes in the datasets provide no help in classification, since very small sets of genes can achieve comparable results. Table 3 shows the gene subsets for the best classification results for the above datasets.

The experimental results also suggest that there may be multiple gene subsets that achieve the best classification accuracy. For example, the MLL Leukemia dataset has more than 100 gene subsets that can achieve maximal classification accuracy.

We also compare our experimental results with the previously published results in [8]. In order to make fair comparisons with [8], we tried to re-produce their experiments by applying both their selected genes and their parameters (RBF kernel with C=32, and g=0.0078125) in LIBSVM on our training and test sets. The results are summarized in Table 4.

Table 2. Classification accuracy after applying BBDFS on 7 microarray datasets

Data Set	Best training Accuracy	Best Test Accuracy	Maximum # of Selected Genes
Lung Carcinoma	100% (Test Accuracy = 99.01%) # Genes = 5	99.01% (Training Accuracy = 100%) # Genes = 5	6 Training Accuracy = 100% Test Accuracy = 97.03%
Brain Cancer	100% (Test Accuracy = 86.21%) # Genes = 2	93.10% (Training Accuracy = 85.71%) # Genes = 2	2 Training Accuracy = 100% Test Accuracy = 86.21%
Prostate Cancer	100% (Test Accuracy = 92.16%) # Genes = 3	96.08% (Training Accuracy = 96.08%) # Genes = 3	3 Training Accuracy = 100% Test Accuracy = 92.16%
CNS	100% (Test Accuracy = 66.67%) # Genes = 3	88.89% (Training Accuracy = 84.85%) #Genes = 3	3 Training Accuracy = 100% Test Accuracy = 66.67%
Colon Cancer	100% (Test Accuracy = 87.10%) # Genes = 2	93.55% (Training Accuracy = 93.55%) # Genes = 3	3 Training Accuracy = 100% Test Accuracy = 87.10%
ALL-MLL Leukemia	100% (Test Accuracy = 100%) # Genes = 4	100% (Training Accuracy = 100%) # Genes = 4	4 Training Accuracy = 100% Test Accuracy = 100%
MLL Leukemia	96.49% (Test Accuracy = 100%) # Genes = 3	100% (Training Accuracy = 96.49%) # Genes = 3	4 Training Accuracy = 96.49% Test Accuracy =100%

Table 3. Best gene subsets found by BBDFS for all 7 datasets

Datasets	*Gene Subset*
Lung Carcinoma	41325_at 36133_at 41231_f_at 41491_s_at 32562_at
Brain Cancer	38421_at 40272_at
Prostate Cancer	36607_at 40282_s_at 37639_at
CNS	L47125_s_at S71824_at U62801_at
Colon Cancer	H51524 Z50753
ALL-MLL Leukemia	M20919_at M62783_at M19645_at M28209_at
MLL Leukemia	36002_at 33412_at 33439_at

Table 4. Experimental results reported in [8] for lung carcinoma, brain, and prostate cancer datasets using genetic algorithms

Data Set	Best Training Accuracy	Best Test Accuracy	Minimum Number of Selected Genes
Lung Carcinoma	100.0% (Test Accuracy = 69.31%) # Genes = 40	69.31% (Training Accuracy = 100.0%) # Genes = 40	40 Training Accuracy = 100.0% Test Accuracy = 69.31%
Brain Cancer	100% (Test Accuracy = 68.97%) # Genes = 13	68.97% (Training Accuracy = 100%) # Genes = 13	3 Training Accuracy = 100% Test Accuracy = 58.62%
Prostate Cancer	100% (Test Accuracy = 92.16%) # Genes = 24	96.08% (Training Accuracy = 98.04%) # Genes = 30	6 Training Accuracy = 98.04% Test Accuracy = 88.24%

These comparisons suggest that our BBDFS method obtains comparable results for the prostate cancer dataset, and BBDFS obtains much better results for both the lung carcinoma and brain cancer datasets. Note that the results here are quite different from the ones in [8], which we could not reproduce. They reported a classification result of 100% training and 90.48% test accuracy, while we could only reproduce 100% training and 68.97% test accuracy in Brain Cancer dataset.

4 Conclusion and Future Work

We have presented a new method for finding minimal sets of informative genes in microarray data. It is essentially a feature selection method that finds minimal sets of significant features using branch-and-bound depth-first search (BBDFS). Based on the successful test results shown here, it appears BBDFS can select very small gene subsets with good classification accuracy, and we believe the method can be helpful with microarray data that has a number of phenotype classifications (multiple classes).

There are potential advantages of algorithms like BBDFS in the analysis of genes potentially related to cancer. Current methods in extracting informative genes often return tens or hundreds of genes, and it can be extremely laborious to analyze each of these. Since our algorithm returns very small sets of genes (on the order of 6 here, with good classification accuracy), this analysis can be more focused and possibly significantly limited.

One drawback of the BBDFS algorithm is in overhead cost, since it evaluates the informativeness (classification ability) of each individual attribute. Another drawback is that if the number of training instances is very large (> 10000, say), the algorithm may become computationally costly because it evaluates attributes by using all training instances. Fortunately, real microarray datasets often involve only hundreds of training instances, so this may not be a drawback for many applications.

Some future work will involve the analysis of the gene subsets found by the BBDFS algorithm. It is important to investigate the biological meaning of gene subsets found by the algorithm. Another future direction will be in generalization of BBDFS to be a more general feature selection method. This will involve designing ways to evaluate attributes that do not use all training instances.

Acknowledgments. We wish to acknowledge Brain Cancer for motivating this work. D.S. Parker was supported by NIH grants 1P20MH065166, 1U54RR021813, and the UCLA Center for Computational Biology (CCB).

References

1. Alon U., Barkai N., Notterman D.A., Gish K., Ybarra S., Mack D., and Levine A.J., "Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays", *Proceedings of National Academy of Sciences,* 96:6745-6750, 1999
2. Armstrong S.A., Staunton J.E., Silverman L.B., Pieters R., den Boer M.L., Minden M.D., Sallan S.E., Lander E.S., Golub T.R., and Korsmeyer S.J., "MLL Translocations Specify A Distinct Gene Expression Profile that Distinguishes A Unique Leukemia", *Nature Genetics*, 30:41-47, January 2002.
3. Bø, T., and Jonassen, I., New feature subset selection procedures for classification of expression profiles. *Genome Biology*, 3(4):research0017.1–0017.11, 2002.
4. Bhattacharjee A et al, "Classification of Human Lung Carcinomas by mRNA Expression Profiling Reveals Distinct Adenocarcinoma Subclasses", *Proceedings of National Academy of Sciences*, volume 98, pages 13790–13795, 2001.
5. Chang C.-C., and Lin, C.-J., "LIBSVM : A Library for Support Vector Machines", 2001. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
6. Golub, T.R., et al. Molecular classifications of cancer: Class discovery and class prediction
7. by gene expression monitoring. *Science,* 286(5439):531–7, 1999.

8. Nutt C.L et al, “Gene Expression-Based Classification of Malignant Gliomas Correlates better with Survival than Histological Classification”, *Cancer Research*, 63(7):1602–1607, 2003.
9. Paul, T.K., and Iba, H., Extraction of Informative Genes from Microarray Data, *Proceedings of the Genetic and Evolutionary Computation Conference*, Washington DC, USA, pp 453-460, 2005.
10. Pomeroy S.L et al, "Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression", Letters to Nature, *Nature*, 415:436-442, January 2002.
11. Singh D et al, “Gene Expression Correlates of Clinical Prostate Cancer Behavior”, *Cancer Cell*, 1(2):203-9, March 2002.
12. Tusher, V.G., Tibshirani, R., and Chu, G., Significance analysis of microarrays applied to the ionizing radiation response. *PNAS*, 98:5116–5121, 2001.

Noise-Based Feature Perturbation as a Selection Method for Microarray Data

Li Chen[1], Dmitry B. Goldgof[1], Lawrence O. Hall[1], and Steven A. Eschrich[2]

[1] Department of Computer Science and Engineering
University of South Florida
{lchen2,goldgof,hall}@csee.usf.edu
[2] Department of Interdisciplinary Oncology
H. Lee Moffitt Cancer Cancer & Research Institute
Univeristy of South Florida
Tampa, FL 33620, USA
EschriS@moffitt.usf.edu

Abstract. DNA microarrays can monitor the expression levels of thousands of genes simultaneously, providing the opportunity for the identification of genes that are differentially expressed across different conditions. Microarray datasets are generally limited to a small number of samples with a large number of gene expressions, therefore feature selection becomes a very important aspect of the microarray classification problem. In this paper, a new feature selection method, feature perturbation by adding noise, is proposed to improve the performance of classification. The experimental results on a benchmark colon cancer dataset indicate that the proposed method can result in more accurate class predictions using a smaller set of features when compared to the SVM-RFE feature selection method.

Keywords: Feature perturbation, microarray gene expression data, gene selection, classification.

1 Introduction

Microarray technology makes it possible to measure thousands of gene expressions simultaneously. One of the major goals of microarray data analysis is the detection of differentially expressed genes across two kinds of tissue samples or samples obtained under two experimental conditions in this high-dimensional gene space. Due to the characteristics of microarray datasets, which usually have a small number of samples and a large number of gene expressions, feature selection methods are quite important to enable microarray classification. A large number of feature selection methods have been proposed to identify subsets of features, thereby reducing the probability of overfitting. In general, classifier-based gene selection can be performed using filter or wrapper approaches. Recent studies have demonstrated that wrapper methods often give satisfactory classification accuracy [1][2] since they evaluate a subset of features based on the

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 237–247, 2007.

performance of a specific classifier. For example, Guyon et al. introduced a recursive feature elimination (RFE) scheme, in which features were eliminated successively according to their influence on a support vector machine (SVM) based evaluation criterion [3]. Some extended wrapper methods have been proposed based on SVMs [4]. In [5], a wrapper feature selection method was proposed which organized the feature combinations in a tree structure and the learning algorithm was used for evaluation and intelligently searching the tree structure to find the optimal subset of features for classification.

In the context of microarray data analysis, different feature selection methods have their own advantages and disadvantages. For example, the SVM-RFE method in [3] has been shown to be an effective gene selection method, however the classifier used in the feature selection procedure in this method is restricted to an SVM. The wrapper feature selection method [5] does the best search in the feature space for classification but the search procedure is very time consuming. In this paper, we propose a new feature selection method, feature perturbation by adding noise, to improve the performance of classification. This method is more general than the SVM-RFE method since it can be applied to any classifier. By using an adaptive feature elimination strategy, the running time can be dramatically shortened which makes the algorithm practical. Experimental results on the benchmark colon cancer dataset show that the proposed method can achieve higher prediction accuracy with a lower number of features (genes) when compared to the SVM-RFE method.

2 Method

2.1 Feature Perturbation

Feature selection methods are generally applied to all features to eliminate some of the original features and retain a minimum subset of features that yield good classification performance. The feature perturbation method proposed in this paper is a classifier-based, top-down feature eliminating method. The assumption of the feature perturbation method is quite straightforward: irrelevant features will have little influence on classification performance when perturbed by noise. Correspondingly, classification performance should be significantly impacted when important features are perturbed by noise. As a result, irrelevant features with little effect on classification under noise are removed and the subset of important features for classification will be retained.

Fig. 1 shows a flowchart of the feature perturbation method. The feature perturbation method starts with a set of training samples X with all features S as the surviving features. The method is an iterative procedure and the features are recursively eliminated. Each iteration consists of three stages. The first stage is training the classifier on the training samples X, based on the current surviving features S. Any classifier can be used to build the classification model. The second stage is called feature ranking, which computes the ranking criteria for the S features. The change of classification performance on the training samples before and after adding noise to each feature is defined as the ranking criterion for this feature. Feature elimination removes the feature set R, which contains the features with the lowest ranks from S. For computational reasons, it will be more efficient to remove several features at a time rather than one. The whole procedure will be repeated until all features are removed. As a result, a ranked feature list will be generated and one can evaluate the selected feature set on the test samples using the resultant classification model.

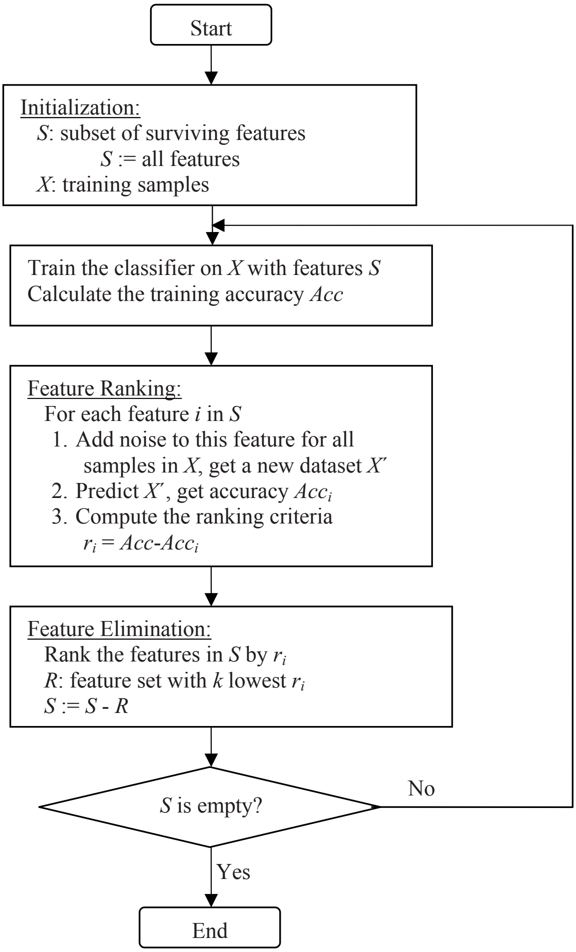

Start
Initialization:
S: subset of surviving features
S := all features
X: training samples
Train the classifier on X with features S
Calculate the training accuracy Acc
Feature Ranking:
For each feature i in S
1. Add noise to this feature for all samples in X, get a new dataset X′
2. Predict X′, get accuracy Acc_i
3. Compute the ranking criteria r_i = Acc-Acc_i
Feature Elimination:
Rank the features in S by r_i
R: feature set with k lowest r_i
S := S - R
S is empty?
No
Yes
End

Fig. 1. Flow chart of feature perturbation method

From the flowchart in Fig. 1 we can see that the most important aspect of the feature perturbation method is the feature ranking procedure. Assume that the active feature set is S and the training sample set is X. The performance of the classifier built on the current dataset is evaluated by the training accuracy represented by Acc. For each feature i in S, noise will be randomly generated and added to this feature for each of the samples in X, to generate a new dataset X'. Class predictions using the new dataset, where the feature i was perturbed by noise, will give us a prediction accuracy denoted as Acc_i. The ranking criterion for feature i can be calculated as the difference between Acc and Acc_i.

In the feature ranking procedure, noise is added to each feature to measure the importance of the features for classification. Here the problem is how to decide the characteristics of the noise. Intuitively, noise should be dependent on each particular feature if we assume that the features are independent of each other. However, it is not necessary for the noise to follow a certain distribution since the noise is only used for perturbing the features. In the following experiments, we use a uniform distribution to randomly sample noise. For each feature i the noise follows a uniform distribution with the parameters related to this feature, which can be represented by following Eq. 1:

$$Noise_i \sim Uniform(-c * sd_i, c * sd_i)\,. \tag{1}$$

where, sd_i is the standard deviation of feature i across all training samples, and c is a constant factor which determines the noise level. Large amounts of noise were required to get the appropriate perturbation of the features in our experiments. This procedure was repeated 5 times in each iteration to get an average ranking criteria for each feature since the noise is randomly generated in the experiments.

Given the characteristics of microarry datasets which usually have small number of samples and a large number of gene expressions, the feature elimination procedure will be time consuming if features are eliminated one by one. Based on the assumption that there are only a few relatively important features for classification and most of the features are irrelevant, we used an adaptive feature elimination strategy in which the number of features removed is determined by the current number of surviving features. For example, half the features can be removed at a time rather than just one when there are likely to be a large number of irrelevant features. When the number of features is less than some threshold, one can eliminate the features individually to get more accurate results.

2.2 Feature Perturbation Method vs. SVM-RFE Method

An alternative algorithm for feature selection is the SVM-RFE method [3]. This approach eliminates the features successively according to their influence on the training SVM model. Fig. 2 shows the flowchart of the SVM-RFE method. Compared to the flowchart of the feature perturbation method in Fig. 1, we can see similarities except for the first two stages in each iteration. In the first stage of SVM-RFE, a SVM-based training model C is built on the training samples X and current surviving features S. In the SVM-RFE feature ranking, the weight vector W is computed based on the training model and the ranking criteria for each feature i is determined by the corresponding weight w_i for this feature.

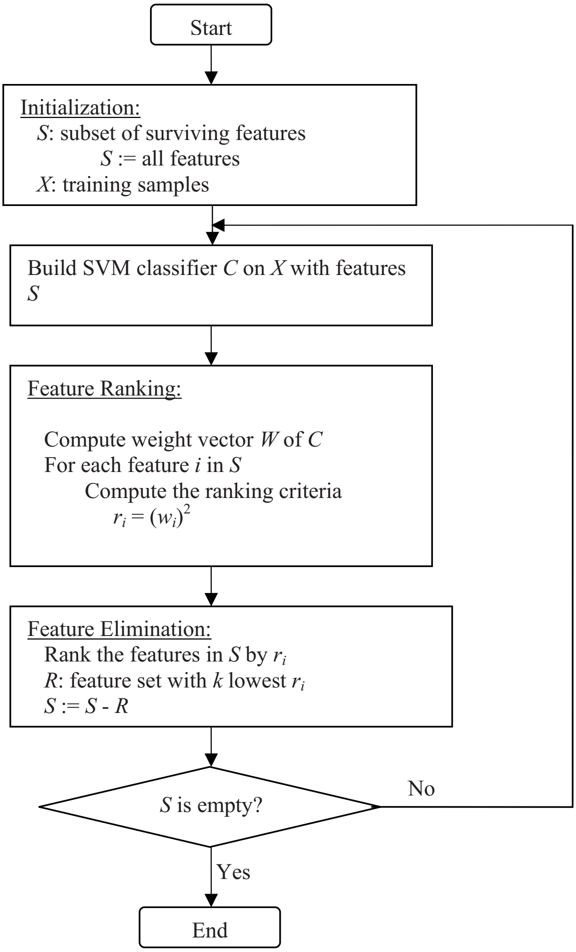

Start
Initialization:
S: subset of surviving features
S := all features
X: training samples
Build SVM classifier C on X with features S
Feature Ranking:
Compute weight vector W of C
For each feature i in S
Compute the ranking criteria
$r_i = (w_i)^2$
Feature Elimination:
Rank the features in S by r_i
R: feature set with k lowest r_i
$S := S - R$
S is empty?
No
Yes
End

Fig. 2. Flow chart of SVM-RFE method

One distinct advantage of the feature perturbation method is that it is applicable to any classifier for feature selection and evaluation, while SVM-RFE is a method based only on support vector machines. Therefore, the feature perturbation method is more general than SVM-RFE. The computational complexity of the feature perturbation method in each feature elimination procedure is $O(d^2 * n)$, while SVM-RFE requires $O(d * n)$ evaluation. Here, d is the number of active features and n is the number of training samples. The feature perturbation method is expected to be slower than the SVM-RFE method but we can use the adaptive feature elimination approach to speed up the computation time. The last characteristic of the feature perturbation method is randomness in selecting the features since the noise is randomly generated and added.

3 Experimental Results

3.1 Data and Parameters

We present results on a well-known benchmark microarray dataset in colon cancer which can be obtained from the website http://microarray.princeton.edu/onc-ology/affydata/. The colon cancer data consists of 62 tissue samples including 22 normal and 40 colon cancer tissues. Each sample has 2000 gene expression values. We use tenfold cross validation to evaluate prediction accuracy between the feature perturbation method and SVM-RFE method. Although the feature perturbation method is applicable to any classifier, support vector machines are used in our experiments in order to compare with the SVM-RFE method. The SVM used as the classifier is a modified version of libSVM [6]. Because no substantial difference in performance was observed in our preliminary classification using SVM with different kernels, a linear kernel with parameter $C = 1$ was used in the following experiments to reduce both training time and the probability of overfitting. The sequential minimal optimization (SMO) algorithm was the optimization algorithm used.

Since the samples are unequally distributed amongst both classes, weighted accuracy will yield a better estimate of how well the classifier performs than total accuracy [7]. Weighted accuracy is defined in Eq. 2:

$$Weighted\ Accuracy = \left(\frac{TP}{TP + FN} + \frac{TN}{FP + TN} \right) /2 \, . \tag{2}$$

where TP, FP, TN, and FN are the number of true positives, false positives, true negatives, and false negatives, respectively in a confusion matrix in the context of prediction with colon cancer, as shown in Table 1.

Table 1. Confusion matrix

	Predicted Cancer	Predicted Normal tissue
Actual Cancer	True Positive (TP)	False Negative (FN)
Actural Normal tissue	False Positive (FP)	True Negative (TN)

In the following experiments, all the results will be reported using weighted accuracy.

All the programs were implemented using the C language on a personal computer with a 2.8 GHZ CPU and 1 GB of RAM.

3.2 Experiments and Results

We have observed that 1522 of the 2000 features have p-values>0.05 in the t-test, which validates our assumption that most of the features (genes) are uninformative for classification. Therefore we apply the following adaptive feature elimination strategy to speed up the run time for both the feature perturbation and the SVM-RFE method in the experiments. Starting from 2000 features, half of the features were removed each time when the number of features was greater than 200, after that, features were removed one by one until none remain. As a result, the evaluation will be performed for the feature sets with the size of 2000, 1000, 500, 250, 125, 124, ..., 1.

The procedure for the experiments was the following. First, as a baseline experiment, the SVM-RFE method was applied to the dataset using the original feature elimination strategy and the adaptive strategy to see if there was a significant difference between them. The dataset was randomly stratified into training and test datasets using a tenfold cross validation approach. The SVM-RFE method was utilized on the training set to recursively eliminate the features and test accuracies were predicted on the test set using the classifier built on the training data with selected features. This was done 10 times, once for each left out fold, and an average accuracy over the ten folds is reported. The whole procedure was repeated five times with different randomly chosen stratified sets of data in order to get more reliable results. The results reported are the average values of the five experiments.

Fig. 3 shows the comparison of average weighted accuracies of the SVM-RFE method using the two different feature elimination strategies across different numbers of features. In this figure, RFE-1 represents the original feature elimination approach where the features were recursively removed one by one. RFE-M represents the adaptive feature elimination approach. It can be seen from the results that there is very little difference in the performance for these two feature elimination approaches (p-value = 0.054 using the wilcoxon test). Note that the adaptive feature elimination approach has lower accuracies with less features than the original one, which indicates that some important features may be removed at the beginning using this approach. However the adaptive feature elimination approach is much faster than the original approach. The average running time across the five individual experiments for RFE-1 was approximately 150 minutes. On the other hand, the average running time across the five individual experiments for RFE-M was approximately 60 minutes. The experiment indicates that we can use the proposed feature elimination approach to speed up the running time while maintaining comparable performance.

The results reported in Fig. 3 are the average values of five individual cross-validation trials. Fig. 4 is a graph of the weighted accuracies for the best and worst cases in these five individual trials in the SVM-RFE-1 method with the original feature elimination approach. For the best one, the weighted accuracy increases slightly when the number of features decreases, but it drops dramatically in the worst trial. Therefore, the average weighted accuracies are steadily lower than the initial ones with 2000 features and they vary smoothly across different number of features after taking the average.

Next, for the feature perturbation method, we tried different parameter values for c in Eq. 1 to see if it could significantly impact on the performance. c was chosen as 1, 2, and 3 respectively. Due to the limitation of running time, only the adaptive feature elimination approach was used in the algorithm. The dataset was randomly stratified as training and test datasets for a tenfold cross

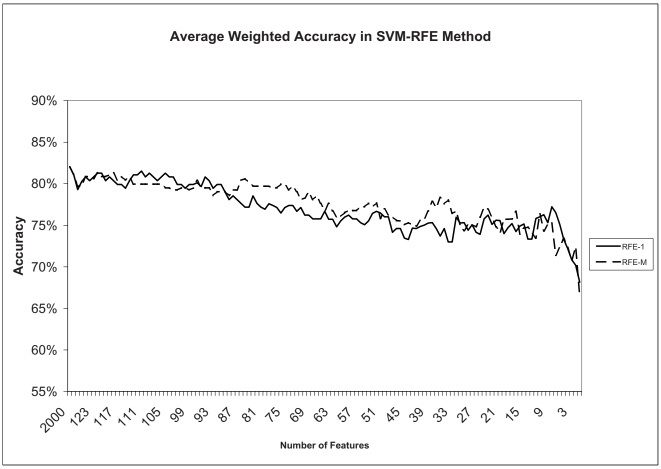


Fig. 3. Comparison of average weighted accuracies of SVM-RFE with different feature elimination approaches at different number of features

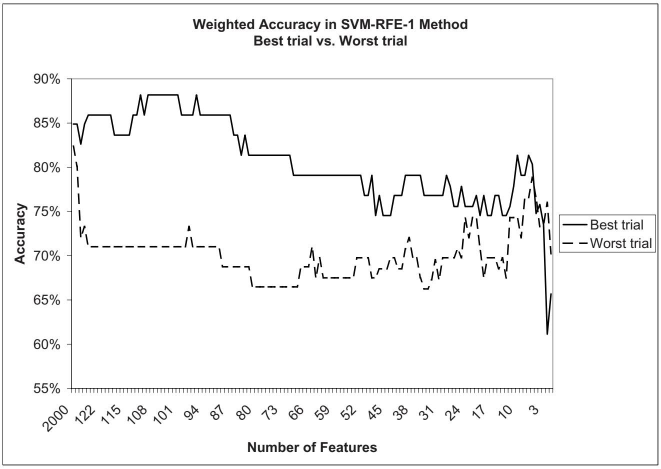


Fig. 4. Weighted accuracies of SVM-RFE in the best and worst trials

validation. The feature perturbation method was implemented on the training dataset to recursively eliminate the features, where the noise was randomly generated from the uniform distribution with the parameter *c*. The SVM classifier was built each time with the current active features and the test dataset with the corresponding active features was predicted to yield the test accuracy. This was done 10 times, once for each left out fold, and and average weighted accuracy over the tenfold was reported. The whole procedure is repeated five times with the same five different randomly chosen stratified sets of data as in the

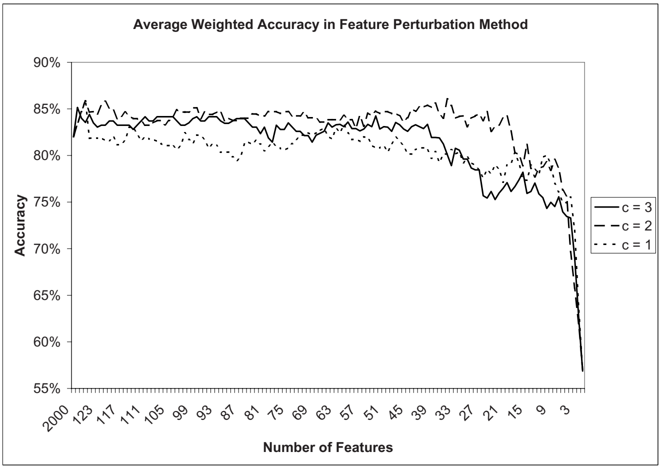


Fig. 5. Average weighted accuracy for feature perturbation method with different parameter values of c at different number of features

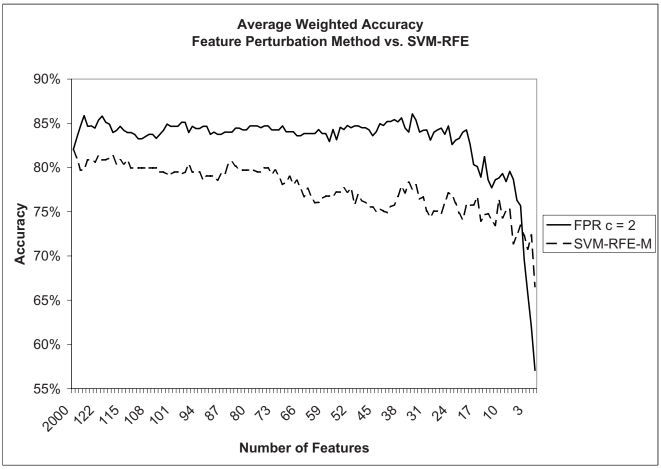


Fig. 6. Comparison of average weighted accuracy between feature perturbation method and SVM-RFE at different number of features

SVM-RFE method in order to get more reliable results. The results reported are the average values of the five experiments for each value of c.

Fig. 5 is a graph of the average weighted accuracies of the feature perturbation method with different parameter c at different number of features. The figure shows that in all cases where c equals 1, 2, and 3, the accuracy is stable or drops slightly with a large number of features and then it decreases dramatically when only a few features are retained. Specifically, the overall performance when $c = 2$

Table 2. Statistical test of performance improvement

Method 1	Method 2	Wilcoxon test p-value
FPR (c=1)	SVM-RFE	$< 2.2e - 16$
FPR (c=2)	SVM-RFE	$< 2.2e - 16$
FPR (c=3)	SVM-RFE	$< 2.2e - 16$

is better than the cases when c = 1 or c = 3. The average weighted accuracy was above 80% for c = 2 when using 15 features, as compared to 75.93% for c = 3 and 77.34% for c = 1. The best average weighted accuracy can be achieved at 86.07% with 34 features when c = 2. The average running time across five individual experiments for the feature perturbation method with the adaptive feature elimination approach was approximately 800 minutes, which is more than 10x slower than the SVM-RFE method.

Fig. 6 shows the comparison of the average weighted accuracies of the feature perturbation method and the SVM-RFE method. The adaptive feature elimination approach was used to remove the features recursively. For the feature perturbation method, we chose the optimal parameter value of c as 2. As can be seen in this figure, the feature perturbation method outperformed SVM-RFE significantly in detecting differentially expressed genes for classifying the colon cancer dataset with the most number of features. The performance of the SVM-RFE method is only better than the feature perturbation method when the number of features (genes) is less than 5, where accuracy drops off dramatically in both approaches.

For a better comparison of the two different feature selection methods, statistical tests were applied to measure the significance of improvements in prediction accuracy. The feature perturbation method with different parameter values of c was compared to the SVM-RFE method. As the normality test fails, the Wilcoxon test [8] was used to test the significance of differences across numbers of features. Listed in Table 2 are the p-values of the Wilcoxon test, comparing the feature perturbation method (FPR) and SVM-RFE. The p-values indicate there is a significant improvement in accuracy using the feature perturbation method, as compared to the SVM-RFE method.

4 Conclusion

This paper presents a noise-based feature perturbation method as a selection method for microarray data. We began with the hypothesis that truly informative genes will cause a classifier to fail when peturbed by noise. We have described a recursive elimination procedure to remove features that change overall classification accuracy the least when modified by noise. This procedure is similar in form to the SVM-RFE method however it is not constrained to the use of support vector machines. The implementation details and computational complexity were described.

The noise-based feature peturbation method was compared to the SVM-FRE method through a series of experiments on the prediction of colon cancer vs. normal colon tissue. Experimental results showed that the feature perturbation method can achieve higher prediction accuracies with fewer number of features than the SVM-RFE method. However, several issues in the feature perturbation method need further investigation. The introduction of noise is based on the variability of the feature, however more work is required in identifying the optimal factor (the *c* parameter) to use. Additional strategies for speedup within the implementation will also be important to address.

Acknowledgments. This research was supported by the Department of Defense, National Functional Genomics Center project, under award number DAMD 17-02-2-0051. Views and opinions of, and endorsements by, the author(s) do not reflect those of the US Army or the Department of Defense.

References

1. Inza, I., Larranaga, P., Blanco, R., Cerrolaza, A.J.: Filter versus wrapper gene selection approaches in DNA microarray domains. Artifical Intelligence Medicine, 31 (2004) 91-103
2. Xiong, H., Swamy, M.N.S., Ahmad, M.O.: Optimziing the Kernel in the Empirical Feature Space. IEEE trans. Neural Networks 16 (2001) 460-474
3. Guyon, I., Weston, J., Barnihill S., Vapnik, V.: Gene Selection for Cancer Classification using Support Vector Machines. Machine Learning, 46 (2002) 389–422
4. Rakotomamonjy, A.: Variable Selection using SVM-based Criteria. Journal of Machine Learning Research. Vol 3. (2003) 1357–1370
5. Kohavi, R., John, G.H.: Wrappers for Feature Subset Selection. Artificial Intelligence Archive,97 (1997) 273–324
6. Chang, C.C., Lin, C.J.: A Library for Support Vector Machines, libsvm. http://www.csie.ntu.edu.tw/-cjlin/libsvm.
7. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kauffman Publishers. 2000
8. Lehmann, E.L.: Nonparametrics:Statistical Methods Based on Ranks. Francisco, CA: Holden-Day. 1975

Efficient Generation of Biologically Relevant Enriched Gene Sets

Igor Trajkovski and Nada Lavrač

Department of Knowledge Technologies, Jozef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
{igor.trajkovski,nada.lavrac}@ijs.si

Abstract. Gene set enrichment analysis is a microarray data analysis method that uses predefined gene sets and ranks of genes to identify significant biological changes in microarray data sets. In this paper we present a novel method integrating gene interaction information with Gene Ontology (GO) for the construction of new interesting enriched gene sets. The experimental results show that the introduced method improves over traditional methods that compute the enrichment of a single GO terms, i.e. that it is capable to find new statistically relevant descriptions of the biology governing the experiments not detectable by the existing methods.

1 Introduction

High-throughput technologies such as DNA microarrays and proteomics are revolutionizing biology and medicine. Global gene expression profiling using microarrays monitors changes in the expression of thousands of genes simultaneously. The large amounts of data acquired must then be reduced or "translated" to a smaller set of genes representing meaningful biological differences between control and test systems and validated in an experimental or clinical setting.

In a typical experiment, mRNA expression profiles are generated for thousands of genes from a collection of samples belonging to one of two classes - for example, tumors that are sensitive vs. those resistant to a drug. The genes can be ordered in a ranked list L, according to the difference of expression between the classes. The challenge is to extract the meaning from this list.

A common approach involves focusing on a handful of genes at the top of L (genes showing the largest difference in its expression between the classes), to extract the underlying biology responsible for the phenotypic differences. This approach has a few major limitations:

– After correcting for multiple hypotheses testing, no individual gene may meet the threshold for statistical significance, because the relevant biological differences are small relative to the noise inherent to the microarray technology.
– The opposite situation, one may be left with a long list of statistically significant genes without any common biological function.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 248–259, 2007.

– Single-gene analysis may miss important effects on pathways. Cellular processes often affect sets of genes acting jointly. An increase of 20% in all genes encoding members of a biological process may dramatically alter the execution of that process, and its impact on other processes, than a 10-fold increase in a single gene.
– It is not a rare case when different groups studying the same biological system, report a list of statistically significant genes from the two studies that have a significantly small overlap.

To overcome these analytical challenges, recently method was developed, called Gene Set Enrichment Analysis (GSEA) [1], that evaluates microarray data at the level of gene sets. Biologically interesting sets of genes, for example genes that belong to a pathway or genes known to have the same biological function, are good examples of such gene sets. The most popular choice for gene sets are genes annotated with some GO term. The goal of GSEA is to determine whether members of a gene set S tend to occur toward the top of the list L, in which case the gene set is correlated with the phenotypic class distinction.

In this work we propose a method for generating new gene sets that have relevant biological interpretations, by combining the existing gene sets, and by inclusion of gene-gene interaction information available from the public gene annotation databases. The experimental results show that our method can find descriptions of interesting enriched gene sets, that traditional methods are unable to discover. We applied the proposed method to three gene expression data sets with the following respective sets of sample classes: (i) acute lymphoblastic leukemia (ALL) vs. acute myeloid leukemia (AML), (ii) seven subtypes of ALL, and (iii) fourteen different types of cancers. Significant number of discovered gene sets have description which highlights the underlying biology that is responsible for distinguishing one class from the other classes.

The paper is organized as follows. In Section 2 we give background information about Gene Ontology and differentially expressed genes. Section 3 provides details of the Gene Set Enrichment Analysis. Section 4 presents the idea of our approach, and the steps taken in the construction of interesting gene sets. Section 5 presents the results of the experiments. In Section 6 we draw some final conclusions.

2 Background

2.1 Gene Ontologies

One of the most important tools for the representation and processing of information about gene products and functions is the Gene Ontology (GO)[1]. It provides a controlled vocabulary for the description of cellular components, molecular functions, and biological processes.

[1] http://www.geneontology.org

As of December 2006, GO contains 1864 cellular component, 7513 molecular function and 12549 biological process terms. Terms are organized in parent-child hierarchies (see Fig. 1), indicating either that one term is more specific than another (is_a) or that the entity denoted by one term is part of the entity denoted by another (part_of). Typically, such associations (or "annotations") are first of all established electronically and later validated by a process of manual verification which requires the annotator to have expertise both in the biology of the genes and gene products and in the structure and content of GO. GO, in spite of its name, is not an ontology in the sense accepted by computer scientists, in that it does not deal with axioms and formalized definitions associated to terms. It is rather a taxonomy, or, as the GO Consortium puts it, a "controlled vocabulary" providing a practically useful framework for keeping track of the biological annotations applied to gene products.

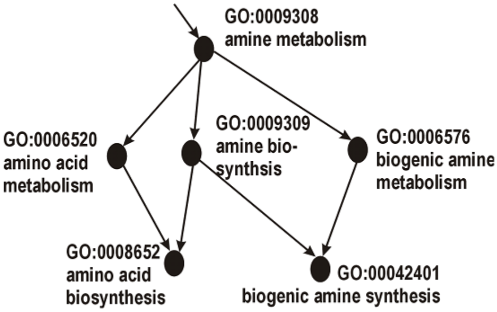


Fig. 1. This figure shows a part of GO providing the annotations concerning amine metabolism

2.2 Differentially Expressed Genes

Differentially expressed genes are genes that are expressed differently (relative. to the reference) between the conditions of interest. In the context of finding differentially expressed genes, the null hypothesis for each gene is that it is not differentially expressed between two conditions, usually against the two-sided alternative hypothesis that the gene is up- or down regulated. The most commonly used statistical test in this setting has been the two-sample t-test [3] [4], although other statistics such as the signal-to-noise ratios [2], or Pearson's correlation [5], have often been used.

Let $T(g,c)$ denote the t-test score of gene g for a target class c, which is computed by the following procedure: $[\mu_1(g), \sigma_1(g)]$ and $[\mu_2(g), \sigma_2(g)]$ denote the means and standard deviations of the logarithm of the expression levels of gene g for the samples in class c and samples in $C \setminus c$, respectively. Also, let $N_1 = |c|$ and $N_2 = |C \setminus c|$. $T(g,c)$ is computed by the following formula:

$$T(g,c) = \frac{\mu_1(g) - \mu_2(g)}{\sqrt{\frac{\sigma_1(g)}{N_1} + \frac{\sigma_2(g)}{N_2}}} \tag{1}$$

which reflects the difference between the classes relative to the standard deviation within the classes. Large values of $|T(g, c)|$ indicate a strong correlation between the gene expression and the class distinction, while the sign of $T(g, c)$ being positive or negative corresponds to g being more highly expressed in class c or in other classes.

3 Gene Set Enrichment Analysis (GSEA)

GSEA considers experiments with gene expression profiles from samples belonging to two classes. First, genes are ranked based on the correlation between their expression and the class distinction by using any suitable metric, for instance computed by signal-to-noise ratios or Pearson's correlation. In our experiments we ranked the genes according to their t-score value.

Given a predefined set of genes S (e.g. genes involved in some specific biological process) and ranked gene list L, the goal of GSEA is to determine whether the members of S are randomly distributed throughout L or primarily found at the top of the list.

There are two major steps in the GSEA method:

1. **Calculation of an Enrichment Score.** Enrichment score (ES) reflects the degree to which a set S is overrepresented at the top of the ranked list L. The score is calculated by walking down the list L, increasing a running-sum statistic when encountering a gene in S and decreasing it when gene is not in S. The magnitude of the increment depends on the size of S and the total number of genes N. The enrichment score is the maximum deviation from zero encountered in the random walk. If $L = (g_1, g_2, ..., g_N)$ is a ranked list of genes, according to their t-score, enrichment score ES is calculated as:

$$Hit(S,i) = \sum_{\substack{g_j \in S \\ 1 \le j \le i}} \frac{1}{|S|} \qquad\qquad Miss(S,i) = \sum_{\substack{g_j \in S \\ 1 \le j \le i}} \frac{1}{N - |S|}$$

$$ES(S) = \max_{1 \le i \le N} |Hit(S,i) - Miss(S,i)| \tag{2}$$

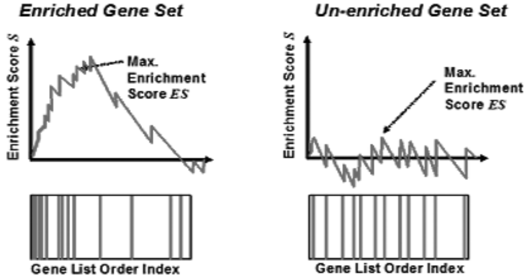


Fig. 2. 'Spectral lines' show the position of gene set members on the ranked gene list

2. **Estimation of Significance Level of ES.** The statistical significance of the ES is computed by using an empirical phenotype-based permutation test procedure that preserves the complex correlation structure of the gene expression data. Specifically, one permutes the phenotype labels and recomputes the ES of the gene set for the permuted data, which generates a null distribution for the ES. The empirical, p-value of the observed ES is then calculated relative to this null distribution. Importantly, the permutation of class labels preserves gene-gene correlations and, thus, provides a more biologically reasonable assessment of significance than would be obtained by permuting genes.

4 Generation of New Gene Sets

Methods that test for enrichment of GO terms have been proposed by [6], [7], [8] and [9]. A comparative study of commonly used tools for analyzing GO term enrichment was presented by [10]. [11] presented two novel algorithms that improve GO term scoring using the underlying GO graph topology.

None of the papers includes the gene interaction information, and none of them presents a method for the construction of novel gene sets, but rather they just calculate the enrichment of an a-priory given list of gene sets.

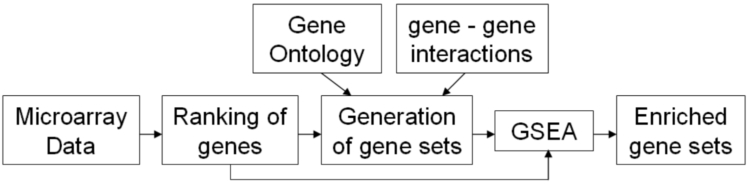


Fig. 3. Data flow of the proposed method for generation of enriched gene sets

First, let us mention some properties of the gene annotations with GO terms:

– one gene can be annotated with several GO terms,
– a GO term may have thousands of genes annotated to it,
– if a gene is annotated with a GO term A then it is annotated with all ancestors of A.

From this information, we can conclude that each GO term defines one gene set, that one gene can be member of several gene sets, and that some gene sets are subsets of other gene sets.

Second, let *Func*, *Proc* and *Comp* denote the sets of gene sets that are defined by the GO terms that are a subterm of the term “molecular function”, “biological process” and “cellular component”, respectively.

Our method relies on two ideas, that are used in the construction of new gene sets:

- **Inclusion of gene interaction information.** There are cases when some abrupted processes are not detectable by the enrichment score, one reason can be that the genes had a slight increase/decrease in their expression, but had a much larger effect on the interacting genes. Therefore we think that it is reasonable to construct gene set whose members interact with another gene set. Gene-gene interaction information is provided for pairs of genes for which there is an evidence that their expression levels are correlated, determined by analysis of microarray data or other experimental methods.

  Formally: if $G_1 \in Func$ (or $Proc, Comp$, respectively), then $G_2 = \{g_2 | g_2$ is a gene, and g_2 interacts with $g_1 \in G_1$ $\}$ was added to $Func$ (or $Proc, Comp$).

- **Intersection of gene sets.** There are cases where two or three given gene sets are not significantly enriched, but their intersection is significantly enriched.

  Formally: if $G_1 \in Func, G_2 \in Proc$ and $G_3 \in Comp$, then $G_4 = G_1 \bigcap G_2 \bigcap G_3$ is a new defined gene set.

  For example, it can happen that gene set defined by the molecular function F is not enriched, because a lot of genes in different parts of the cell execute it and one can not expect that all of them will be over/under expressed, but if genes with that function in some specific part of the cell C_{part} are abnormally active, then it can be elegantly captured by the following gene set:

  $$\text{Func(F)} \bigcap \text{Comp}(C_{part}).$$

  Note that all genes are annotated with the top three GO terms: “molecular function”, “biological process” and “cellular component”, which means that top three GO terms contain all the genes.

The newly defined gene sets are interpreted very intuitively. For example, the gene set defined as intersection of a “functional” term A and “process” term B:

$$\text{function(A), process(B)} \equiv \text{Func(A)} \bigcap \text{Proc(B)}$$

is interpreted as: *Genes that are part of the process B and have function A*, or:

$$\text{int(process(A))} \equiv \{g_2 | g_2 \text{ is a gene, and } g_2 \text{ interacts with } g_1 \in \text{Proc(A)}\}$$

is interpreted as: *Genes that interact with genes that are members of process A*, or:

$$\text{int(process(A),component(B))} \equiv \text{int(Proc(A))} \bigcap \text{int(Comp(B))}$$

is interpreted as: *Genes that interact with genes that are members of process A, and genes that operate in cellular component B.*

The number of the newly defined gene sets is huge. In December 2006, $|Func|$ = 7513, $|Proc|$ = 12549 and $|Comp|$ = 1846. After the inclusion of the gene interaction information, the size of these sets is doubled. Then the number of newly generated gene sets is:

$$2^3 \times |Func| \times |Proc| \times |Comp| \approx 1.4 \times 10^{12}$$

For each of these sets we need to compute its enrichment score, ES, that takes linear time in the number of genes ($\approx 2 \times 10^4$), we get $\approx 3 \times 10^{16}$ floating operations. If we want to statistically validate founded enriched gene sets, usually with 1000 permutation tests, we get $\approx 10^{20}$ operations, that is well above the average performance of today PC's. Therefore we need to efficiently search the space of newly generated gene sets for possible enriched gene sets.

The first idea for improvement is that we are not interested in generating all possible gene sets, but only those that are potentially enriched, and have some minimum number of genes at the top of the list, for example 5 in the first 100, or 10 in the first 300 genes of the list (this was the constraint used in our experiments). That is a weak constraint concerning the biological interpretation of the results, because we are not really interested in the gene sets that do not have this number of genes at the top of the list, but it is a hard constraint concerning the pruning of the search space of all gene sets. By having this constraint we can use the GO topology to efficiently generate all gene sets that satisfy it.

GO is a directed acyclic graph, the root of the graph is the most general term, which means that if one term (gene set) does not satisfy our constraint, than all its descendants will also not satisfy it, because they cover a subset of the genes covered by the given term. In this way we can significantly prune the search space of possible enriched gene sets. Therefor, we first try to construct gene sets from the top nodes of the GO, and if we fail we do not refine the last added term that did not satisfy our constraint.

After the construction of the gene sets that satisfy our constraint, we calculate their ES value, and statistically validate this values using the permutation testing.

In the original version of Kolmogorov-Smirnov test, used by GSEA, the ES statistic used equal weights at every step, which yielded high scores for sets clustered near the middle of the ranked list. These sets do not represent biologically relevant correlation with the phenotype. We addressed this issue by weighting the steps according to each genes correlation with a phenotype. Like in the original version we first rank the N genes to form L = $(g_1, g_2, \ldots, g_N)$ according to their t-score, $t(g_j) = t_j$, of their expression profiles with class c. Then the running sum Hit is computed by following formula:

$$Hit(S,i) = \sum_{\substack{g_j \in S \\ 1 \leq j \leq i}} \frac{|t_j|}{\sum_{g_j \in S} |t_j|}$$

The other running sum, $Miss$, and ES statistic were calculated with the original formulas.

5 Experiments

We applied the proposed methodology to three classification problems from gene expression data, with the aim to describe the most important biological processes that are responsible for class differentiation.

The first problem was introduced in [2] and aims at distinguishing between samples of ALL and AML from gene expression profiles. The second problem was described in [12] and aims at distinguishing different subtypes of ALL (6 recognized subtypes plus a separate class 'other' containing the remaining samples). The third problem was defined in [13]. Here one tries to distinguish among 14 classes of cancers from gene expression profiles. Gene annotations and interaction data was downloaded from Entrez database ftp://ftp.ncbi.nlm.nih.gov/gene/.

Note that this paper does not address the learning task of discriminating between the classes. Instead, for the given target class we aim at finding relevant enriched gene sets that can capture the underlying biology characteristic for that class.

Table 1. Some of the enriched gene sets in the first dataset, with p-value ≤ 0.001

Class	Gene Set	ES
ALL	1. int(Func('zinc ion binding'), Comp('chromosomal part'), Proc('interphase of mitotic cell cycle'))	0.60
	2. Proc('DNA metabolism')	0.59
	3. int(Func('RNA polymerase II transcription factor activity'), Proc('ubiquitin cycle'), Comp('intracellular non-membrane-bound organelle'))	0.56
	4. int(Func('ATP binding'), Comp('chromosomal part'), Proc('DNA replication'))	0.55
AML	1. int(Func('metal ion binding'), Comp('cell surface'), Proc('response to pest, pathogen or parasite'))	0.54
	2. int(Comp('lysosome'))	0.53
	3. Proc('inflammatory response')	0.51
	4. int(Proc('inflammatory response'), Comp('cell surface'))	0.51

5.1 Experimental Results

To illustrate the straightforward interpretability of the enriched gene sets found by our approach, we provide the best-scoring gene sets for some of the target classes in the mentioned three classification problems (see Table 1, 3 and 4). We should mention that enriched gene sets that include too general GO terms (i.e. "biological function", "protein binding", "cellular physiological process", "cytoplasm", etc.), were removed from the result list.

For comparison of enrichment of the found gene sets with the gene sets defined by a single GO term, in Table 2 we list the most enriched gene sets defined by a single GO term, for the first dataset. We can see that ES of the single GO terms is much smaller then the ES of the newly constructed gene sets, and most

Table 2. Summary of GSEA results for the first dataset, with p-value ≤ 0.005. Gene sets constructed from a single GO term.

Class	Gene Set	ES
ALL	1. Proc('DNA metabolism')	0.59
	2. Comp('intracellular non-membrane-bound organelle')	0.35
	3. Proc('development')	0.22
	4. Comp('cytoplasmic part')	0.22
	4. Proc('transport')	0.22
AML	1. Proc('inflammatory response')	0.51
	2. Proc('response to chemical stimulus')	0.41
	3. Proc('proteolysis')	0.38
	4. Proc('cell communication')	0.33

Table 3. Some of the enriched gene sets in the second data set, with p-value ≤ 0.001

CLASS	Gene Set	ES
BRC	1. Proc('cell adhesion'),Comp('integral to membrane')	0.56
	2. int(Func('zinc ion binding'), Proc('cell surface receptor linked signal transd.'), Comp('endoplasmic reticulum'))	0.54
	3. int(Func('metal ion binding'), Proc('cell migration'),Comp('membrane'))	0.53
E2A	1. int(Func('calc. ion bind.'), Proc('protein kinase casc.'), Comp('intracellular membrane-bound organelle'))	0.57
	2. Proc('cell adhesion'),Comp('membrane')	0.57
	3. int(Func('ATP binding'), Comp('integral to membrane'), Proc('reg. of transcr. from RNA poly. II promoter'))	0.55
MLL	1. int(Func('protein kinase activity'), Comp('mem. fraction'), Proc('transmem.rec.protein.tyros.kinase.sig.path.'))	0.63
	2. int(Func('SH3/SH2 adaptor activity'), Proc('apoptosis'), Comp('cytoskeleton'))	0.61
T_ALL	1. int(Func('protein-tyrosine kinase activity'), Proc('positive regulation of T cell proliferation'), Comp('immunological synapse'))	0.92
	2. Proc('antigen presentation'), Comp('integral to membrane')	0.88
TEL	1. int(Proc('cell adhesion'), Comp('cell junction'))	0.65
	2. int(Proc('synaptic transmission'), Comp('cytoskeleton'))	0.57

importantly, the found gene sets are constructed from not enriched GO terms. Similar results we got for the other two datasets.

5.2 Statistical Validation

The following procedure calculated the significance of an observed ES by comparing it with the set of scores ES_{NULL} computed with randomly assigned phenotypes:

1. Randomly assign the original phenotype labels to samples, reorder genes according to their t-score values, and re-compute ES(S).
2. Repeat step 1 for 1,000 permutations, and create a histogram of the corresponding **maximum** enrichment scores ES_{NULL}.
3. Estimate the p-value for the ES value of the gene set S from ES_{NULL} by using the histogram computed at step 2. If there was not a case where random labeling of the examples give bigger ES value, then p-value < 0.001.

We use class labeled permutation because it preserves gene-gene correlations and, thus, provides a more biologically reasonable assessment of significance than would be obtained by permuting genes. Importantly, the permutation of class labels preserves gene-gene correlations and, thus, provides a more biologically reasonable assessment of significance than would be obtained by permuting genes.

Table 4. Some of the enriched gene sets in the third data set, with p-value ≤ 0.001

CLASS	GENE SET	ES
BREAST	1. `Func('RNA binding')`	1.03
	2. `int(Func('zinc ion binding'),Comp('nuclear part'))`	0.79
	3. `int(Func('RNA.polym.II.trans.fact.act.'),Comp('nucleus'), Proc('reg. of transcr. from RNA poly. II promoter'))`	0.76
CNS	1. `Func('struc.const.of.ribosome'),Proc('protein biosynth.')`	2.06
	2. `int(Func('actin binding'),Comp('cytoskeletal part'))`	1.33
COLO.	1. `int(Comp('extracellular matrix (sensu Metazoa)'))`	0.62
LYMPH.	1. `int(Func('transmem.rec.activ.'),Comp('int.to.plasma.mem.') Proc('posit.reg. of I-kappaB kinase/NF-kappaB casc.'))`	1.78
MELAN.	1. `int(Func('transcription cofactor activity'), Proc('muscle development'), Comp('nucleus'))`	1.20
MET	1. `int(Proc('MAPKKK cascade'),Comp('membrane'))`	0.45
OVARY	1. `int(Func('zinc ion binding'),Comp('membrane fraction') Proc('phosphorylation'))`	0.45
	2. `int(Func('zinc ion binding'),Comp('integral to membrane') Proc('cell growth'))`	0.42
PANCR.	1. `Proc('proteolysis')`	0.51
	2. `Comp('ribonucleoprotein complex')`	0.50
PROST.	1. `int(Func('androgen receptor binding'),Comp('nucleus'), Proc('reg. of transcr. from RNA poly. II promoter'))`	0.50
	2. `Comp('cytoskeletal part')`	0.49
RENAL	1. `int(Proc('insulin receptor signaling pathway'), Comp('intracellular membrane-bound organelle'))`	0.43
	2. `int(Func('protein-tyr. kinase activ.'),Comp('cyto. part') Proc('regulation of cell growth'))`	0.43

6 Conclusion

We addressed the problem of finding enriched functional groups of genes based on gene expression data. We proposed a novel method for integrating the gene interaction information into the construction of new interesting relevant gene sets. The experimental results show that the introduced method improves over existing methods, and we base our conclusion on the following facts:

– ES of the newly constructed sets are higher then the ES of any single GO terms.
– Newly constructed sets are composed of non-enriched GO terms, which means that we are extracting additional biological knowledge that can not be found by single GO term GSEA.
– This method is generalization of the traditional methods. If we turn-off gene-gene interactions and combination of GO terms, we will get classical single GO term GSEA.

We believe that the strength of the proposed method will be even bigger through the expected increase in both the quality and quantity of gene annotations and gene-gene interaction information in the near future.

Acknowledgment

This research was supported by the Slovenian Ministry of Higher Education, Science and Technology.

References

1. Subramanian A., et al. (2005) Gene set enrichment analysis: A knowledgebased approach for interpreting genome-wide expression profiles. Proc. Natl. Acad. Sci. of the U.S.A., 102(43):15545-15550.
2. Golub T. R., Slonim D. K., Tamayo P., Huard C., Gaasenbeek M. et al. (1999). Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science, 286:5439, 531-537.
3. Snedecor G. W. and Cochran W. G. (1989) Statistical Methods, Eighth Edition, Iowa State University Press.
4. Tsai C. A., Chen Y. J. and Chen J. J. (2003) Testing for differentially expressed genes with microarray data. Nucleic Acids Res 31, e52.
5. Troyanskaya O. G., et al. (2002) Nonparametric methods for identifying differentially expressed genes in microarray data. Bioinformatics 18(11):1454-61.
6. Draghici S., , et al. (2003) Global functional profiling of gene expression. Genomics, 81:98-104.
7. Zeeberg B. R., et al. (2003) GoMiner: a resource for biological interpretation of genomic and proteomic data. Genome Biology, 4(4):R28.
8. Al-Shahrour F., et al. (2004) FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes. Bioinformatics, 20:578-580.
9. Beissbarth T. and Speed T. (2004) GOstat: Find statistically overrepresented Gene Ontologies within a group of genes. Bioinformatics, 1(1):1-2.

10. Khatri P. and Draghici S. (2005) Ontological analysis of gene expression data: current tools, limitations, and open problems. Bioinformatics, 21(18):3587-3595.
11. Alexa A., et al. (2006) Improved Scoring of Functional Groups from Gene Expression Data by Decorrelating GO Graph Structure. Bioinformatics, 22(13):1600-1607.
12. Ross M. E., et al. (2003) Classification of pediatric acute lymphoblastic leukemia by gene expression profile. BLOOD, pp. 2951-2959.
13. Ramaswamy S., et al. (2001) Multiclass cancer diagnosis using tumor gene expression signatures. Proc. Natl. Acad. Sci. of the U.S.A. 18;98(26):15149-54.

Space and Time Efficient Algorithms to Discover Endogenous RNAi Patterns in Complete Genome Data

Sudha Balla and Sanguthevar Rajasekaran

University of Connecticut, Storrs CT 06269-2155, USA

Abstract. RNAi, short for RNA Interference, a phenomenon of inhibiting the expression of genes, is widely adopted in laboratories for the study of pathways and determination of gene function. Recent studies have shown that RNAi could be used as an approach to treat diseases like cancers and some genetic disorders in which the down-regulation of a protein could prevent or stop progression of the disease. In [7], the problem of detecting endogenous dsRNA control elements and their corresponding mRNA target, i.e., the gene under RNAi control by degradation, in complete genomes of species using a suffix tree data structure is discussed. While the algorithm identifies triple repeats in the genome sequence in linear time, its very high memory requirement (12 GB for the *C. elegans* genome of size 100 Mbp) becomes a bottleneck for processing genomes of higher order. In this paper, we give algorithms that are space and time efficient in practice than the suffix tree based algorithm. Our algorithms are based on simple array data structures and adopt basic sorting techniques to identify the desired patterns in a given genome sequence. We achieve a speedup of 23 and reduction in memory requirement by a factor of 12 for the *C. elegans* genome, over the suffix tree approach, making the processing of higher order genomes possible for detecting such endogenous controls and targets for RNAi by degradation.

1 Introduction

RNA Interference or RNAi ([5]) is a phenomenon that inhibits the expression of target genes by the introduction of double-stranded RNA (dsRNA) molecules into the cells of organisms. RNAi has become a widely adopted technique in laboratories to study pathways and determine gene functions in various species. Recent studies show that it could be adopted as a therapy to treat diseases like cancers and genetic disorders in which the mutant gene responsible for the initiation and progression of such disorders is targeted and suppressed [3]. The dsRNA molecules, either synthetic (*in vitro*) or those synthesized *in vivo* as a hairpin loop, are cut into fragments 21-23 nt long (short-interference RNA or siRNA) by a Dicer enzyme present in the cell. These siRNAs associate themselves to RNA Induced Silencing Complex (RISC) and eventually become single stranded. Then, the RISC identifies the substring of the target mRNA that is *anti sense* to one of the two strands of the siRNA attached to it, binds to the

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 260–269, 2007.

mRNA and cleaves it into two near the center of the siRNA strand. The cell identifies the split mRNA as unwanted material and destroys it. Thus, the protein that would be translated from the mRNA will not be produced and the silencing of the gene responsible for the production of the protein is achieved. This process is called *RNAi by degradation. RNAi by inhibition* is another process where micro RNAs (miRNA) approximately 22 nt long bind to sites within the 3' UnTranslated Region (UTR) of the target mRNA and prevent its translation into the corresponding protein ([8]). For a detailed treatment of RNAi please refer to [1]. In *RNAi by inhibition*, perfect matching between the miRNA and the mRNA target site is not necessary but for *RNAi by degradation*, an exact matching is necessary between the siRNA strand and the substring of the target mRNA. In [7], the problem of detecting endogenous dsRNA control elements and their corresponding mRNA target for *RNAi by degradation* in genome sequences is discussed. In this case, the dsRNA control element is a Stem-Loop-Stem (hpRNA) structure formed *in vivo* by the occurrence of two substrings 20-25 nt long, complementary to one another within a small distance along the genome sequence and a third occurrence, which is part of the target gene, that is either one of the above two occurrences that is anywhere in the genome(Figure 1). The first phase is of detecting all such triple repeats in a genome sequence and an algorithm based on a suffix tree data structure is given to detect triplets of at least 20 nt length in [7].

The suffix tree approach is very ideal for applications where several patterns will be searched for matches in a sequence after its suffix tree is built. However,

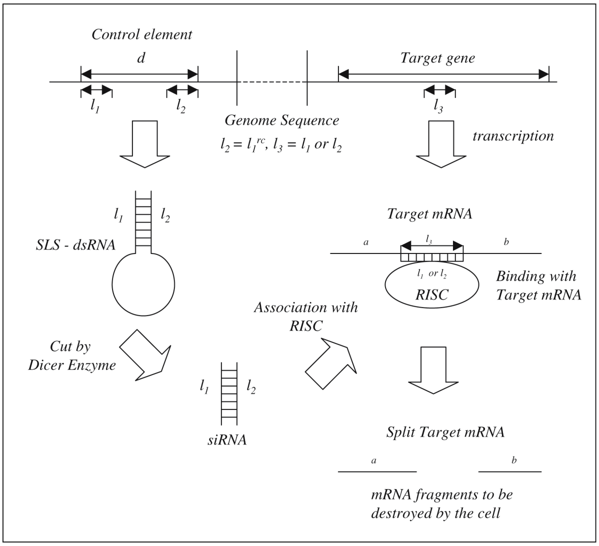


Fig. 1. Endogenous RNAi by Degradation Process

for the above mentioned goal of identifying triplets in a genome sequence, where the lengths of all the triplets are within a very small range, say 20-25 nt, suffix trees may not be appropriate. One of the reasons is that the length of the genome sequence is very large and hence paging could become a very serious issue if the entire tree does not reside in the main memory. This is evident from [7] where the authors report a memory requirement of 12 GB for a genome (*C. elegans*) of size 100 Mb. The time required to process is mentioned as 4 hours on a single processor.

In this paper, we propose two algorithms, *CaTScan1* and *CaTScan2* (for Control And Target Scan), that are based on simple array data structures and adopt basic sorting techniques to identify the triplets. Implementation results of both the algorithms show better performance in practice in space as well as time when compared to the suffix tree algorithm. We have run our algorithms on a PowerEdge 2600 Linux server with 4 GB of RAM and dual 2.8 GHz Intel Xeon CPUs. We have employed only one of these CPUs to process the *C. elegans* genome to identify triplets of length 21 nt. *CaTScan1* takes about eight minutes and no more than 2.5 GB of memory, while *CaTScan2* takes about eleven minutes and no more than 1 GB of memory. Thus our algorithms achieve a speedup of 30 and 23 respectively, while reducing the memory requirement by a factor of 4.8 and 12 respectively, over the suffix tree approach.

2 Problem Description

In [7], the first step in the problem of detecting endogenous dsRNA control elements and their corresponding mRNA target for *RNAi by degradation* involves identifying triple repeats in a given genome sequence. Two of these repeats should be of length 20-25 nt each, be complementary to one another, and occur within a small distance along the genome sequence. A third occurrence, which is a part of the target gene, must be either one of the above two occurrences and can be anywhere in the genome. Formally, the problem is stated as follows:

The Triple Repeat Identification Problem (TRIP): Input are a sequence $S = s_1 s_2 ... s_n$ from an alphabet Σ, and integers l and d. For each element of Σ, a member of this alphabet is defined to be its complement. If L is an l-mer (i.e., a substring of length of l) of S, let L^{rc} stand for the reverse complement of L. The goal is to output every l-mer, L, of S if L^{rc} occurs within a distance d of L in S, and either L or L^{rc} occurs one more time in S.

A brief description of the suffix tree based algorithm is as follows: A suffix tree is built for the given genome sequence S and its reverse complement S^{rc}. A *pre-order traversal* is performed on the tree to calculate the lengths of possible substrings. A *post-order traversal* of the tree is performed in order to calculate the number of occurrences of each such substring in the genome. From these two data, the occurrences of substrings of length at least 20 nt that satisfy the conditions of being a triple repeat are identified. The construction of the tree and each of the two traversals take $O(n)$ time where $|S| = n$.

Exact matching using a suffix tree as the data structure can be done with a preprocessing time of $O(n)$ for a sequence of length n and a search time of $O(l)$ for each pattern of length l thereafter. But, sometimes suffix trees become impractical because of the following vital implementation bottlenecks: (1) If an array of size $\Theta(|\Sigma|)$ is used at each node of the tree, although it achieves constant-time access, the space required will be $\Theta(|\Sigma|n)$; (2) To avoid this, if a linked-list is used at each node, the time taken to search at each node will be $O(|\Sigma|)$; and (3) For very large values of n as in the case of a genome sequence, where the tree would not fit in the memory, paging could become a very serious issue as a suffix tree does not possess good locality properties. For a detailed discussion of suffix trees refer to [6].

Our approach involves creating a collection of all the l-mers and their corresponding reverse complements in the genome sequence, sorting them lexicographically and scanning through the sorted collection to identify the triplets of interest. There are many ways to sort such a collection. For instance, it can be sorted using any comparison-based algorithm such as quick sort, merge sort, etc. Another way is to employ radix sort. In our problem, if we represent every l-mer in the collection as an integer value, we know that the values of elements in the collection lie strictly in the range $[0,\ |\Sigma|^l]$, (i.e.) every element in the collection is a $(\log_2|\Sigma|)l$-bit number. Radix sort sorts the elements with respect to some number of bits at a time starting from the LSBs. For example, these elements can be sorted with respect to w bits at a time, where w is the word length of the computer. In this case, the elements in the collection can be sorted in $(\log_2|\Sigma|)l/w$ phases.

3 New Algorithms for TRIP

The genome sequence S is from the alphabet $\Sigma = \{a, c, t, g\}$. In this alphabet g is the complement of c and a is the complement of t. We map the elements of Σ into integers as follows: $a = 0$, $c = 1$, $t = 2$ and $g = 3$. Thus we need two bits to represent each member of Σ and an l-mer can be represented by a $2l$-bit number.

Algorithm *CaTScan1* adopts the radix sorting approach as follows. Let C be a collection of elements of the form $e = (p, o, v)$, holding the positional (p), orientational (o) and value (v) information of l-mers in S. For every l-mer l_i starting at position i, $1 \le i \le (n - l + 1)$, in S, $e_i^f = (p_i^f, o_i^f, v_i^f)$ and $e_i^{rc} = (p_i^{rc}, o_i^{rc}, v_i^{rc})$ are the two elements representing itself and its reverse complement respectively in C, such that, $p_i^f = p_i^{rc} = i$, $o_i^f = 0$, $o_i^{rc} = 1$, $v_i^f = 2l$-bit number of l_i and $v_i^{rc} = 2l$-bit number of l_i^{rc}. Elements of C are sorted with respect to the integer values of their corresponding l-mers using radix sort. Algorithms that adopt a similar strategy for identifying motifs in a given set of sequences can be found in ([9] and [10]). A scan of the sorted collection C will be sufficient to identify the desired triplets and output them.

Analysis of the above algorithm is straight forward and simple. The collection C could be built in time $O(n + l)$ as follows: The values of both v_i^f and v_i^{rc},

$1 \le i \le (n-l+1)$ of every l-mer in the input sequence S are generated simultaneously. The integer values for the first l-mer, v_1^f and v_1^{rc} are calculated in l-steps, while the integer values for v_i^f and v_i^{rc} for all l-mers, $1 < i \le (n-l+1)$ are calculated in constant time per l-mer as follows.

```
if($i = 1$) {
  $v_i^f = 0$;
  $v_i^{rc} = 0$;
  for $position = 1$ to $l$ do {
      $v_i^f <<= 2$;
      $v_i^f += S[position]$;
      $v_i^{rc} += ((S[position] + 2) mod 4) * 4^{(position-1)}$;
  }
}
else{
    $v_i^f = v_{i-1}^f$;
    $v_i^f -= (S[i-1] * 4^{(l-1)})$;
    $v_i^f <<= 2$;
    $v_i^f += S[i+l-1]$;
    $v_i^{rc} = v_{i-1}^{rc}$;
    $v_i^{rc} >>= 2$;
    $v_i^{rc} += ((S[i+l-1] + 2) mod 4) * 4^{(l-1)}$;
}
```

There are $O(n)$ elements in C and hence the sorting and the scanning steps take time $O(nl/w)$ and $O(n)$ time respectively, where w is the word length of the computer. The space complexity of the algorithm is $O(nl)$ as each element of C are two integers for p_i and v_i (a $2l$-bit integer) and a bit value (0 or 1) for o_i. Therefore, algorithm *CaTScan1* runs in time $O(nl/w)$ using $O(nl)$ space. This algorithm has been implemented and the experimental results are discussed in the next section.

For very large genomes, the memory required by *CaTScan1* could become a bottleneck as it involves holding the values of each v_i^f and v_i^{rc}, two $2l$-bit integers in memory in addition to position i and and its two orientations 0 and 1. In an effort to further reduce the memory requirement of *CaTScan1*, we propose here another algorithm *CaTScan2* that adopts a two phase approach to identify the patterns of interest.

Algorithm *CaTScan2* employs a combination of MSBs first and LSBs first integer sorting. Let k be any integer, $1 \le k \le l$. In the first phase, we partition the l-mers and their corresponding reverse complements of S into 4^k parts (as $|\Sigma| = 4$), with respect to the value of the first k symbols. In particular, two l-mers will be in the same part if their first k symbols are the same. One way of partitioning is as follows. Let $A[1:4^k]$ be an array of linked lists. Scan through S once. For each position i in S, we can think of $s_i s_{i+1} ... s_{i+k-1}$ as a $2k$-bit integer. Let v_i^f be this integer and v_i^{rc} the value of the reverse complement. We add the tuple $(i, 0)$ to the list $A[v_i^f]$ and $(i, 1)$ to the list $A[v_i^{rc}]$.

Now we have at most 4^k *independent* sorting subproblems (one for each list of the array A). Each list of A can also be sorted in a similar way. Consider any list $A[q]$. We can sort elements of $A[q]$ with respect to their next k symbols, and so on. When the size of a list is small ($< 4^k$, for instance), it can be easily sorted using any algorithm. The memory used by this algorithm is $O(n)$ and is not a function of l. In particular, space is reused in solving the independent subproblems.

Alternatively, after the above first phase we can go through each list of A and perform a sorting (with respect to the last $(l-k)$ symbols of the corresponding l-mers) using LSBs first sorting, for example. The advantage of the first phase is very clear. There are nearly $2n$ l-mers and their reverse complements in S. Assume that each symbol of S is picked uniformly randomly from the alphabet Σ. Also assume that the l-mers are independent (which is clearly false since the l-mers could be overlapping). An analysis making this assumption has been proved to hold well in practice (see e.g., [2]). Then the expected size of each list of A is $2n/4^k$. Using Chernoff bounds we can show that the size of each list is no more than $(1+\epsilon)2n/4^k$ with high probability, for any fixed $\epsilon > 0$.

If cln is the amount of memory employed by algorithm *CaTScan1*, then with *CaTScan2*, the space occupied by A is no more than $16n$ (considering that each i is a 32-bit integer; there are n positions on S and $2n$ entries in the linked lists of A; each entry in the linked list is an i and a reference to the next element in the list, thus requiring $4 * 2n * 2 = 16n$ bytes space). The space used to process each list of A is no more than $cl(1+\epsilon)2n/4^k$ with high probability and can be reused for the different lists of A. As a result, the memory used by the new algorithm is $16n + cl(1+\epsilon)2n/4^k$ with high probability (where the probability is over the space of all possible inputs). An example value for k is 6.

Also, the memory requirement of *CaTScan2* could further be reduced (to nearly $8n + cl(1+\epsilon)2n/4^k$) if the lists of A are realized as an array of 4^k arrays whose initial size is calculated by an additional pre-scan of the sequence S. We have implemented algorithm *CaTScan2* using this methodology and provide its performance details in the following section.

The analysis of algorithm *CaTScan2* is as follows: The pre-scan of the genome sequence to calculate the size of each of the 4^k arrays takes $O(n+k)$ time. The partitioning phase takes $O(n+k)$ time. The sorting phase of each of the 4^k arrays with respect to $(l-k)$ symbols of the l-mers takes $O(n_j(l-k)) + O(n_j(l-k)/w)$ time, where n_j is the size of partition j, $1 \leq j \leq 4^k$. Note that $\sum_{j=1}^{4^k} n_j = (2n - l+1)$. Therefore, the time complexity of algorithm *CaTScan2* is $O(4^k + n(l-k))$.

4 Experimental Results

We have implemented Algorithms *CaTScan1* and *CaTScan2* described above as C programs and tested their performance on the genome of *C. elegans* as was performed by [7] and on the genome of *Drosophila melanogaster*. As it can be seen, our prime effort here is focused only on the first phase of identifying RNAi control and target candidates in genome data. We propose here alternative approaches to that based on the suffix tree data structure that are both time and space efficient in practice. Therefore, we do not adopt any additional filtering approaches like those discussed in [7] to further narrow down toward candidates of higher biological significance. There are various research efforts that have focused on identifying the characteristics of the siRNA that achieve RNAi by degradation (see e.g., [3] and [4]). We do not consider any such rules to identify the triple repeats in this research.

The algorithms were run on a PowerEdge 2600 Linux server with 4 GB of RAM and dual 2.8 GHz Intel Xeon CPUs - only one of which is used by both the sequential algorithms. Our program *CaTScan1* identified the triplets in the *C. elegans* genome in 495.27 seconds (just above eight minutes) for the input parameters of $l = 21$ (considered ideal for siRNAs) and $d = 1000$, using only 2.5 GB of memory. Algorithm *CaTScan2* identified the triplets in 663.32 seconds (11 minutes) for the input parameters of $l = 21$ and $d = 1000$, using only 1 GB of memory. Thus our algorithms reflect a speedup of 30 and 23, and reduction in memory requirement by a factor of 4.8 and 12 respectively, over the suffix tree approach. Table 1 shows the performance of *CaTScan1* and *CaTScan2* on the genomes *C. elegans* and *D. melanogaster* respectively.

Further Speedup Strategy: The current version of the algorithms identify triple repeats of a given length l. Therefore, to identify patterns of lengths in a specified range $[l_{min}, l_{max}]$, as would be desired in the case of RNAi control and target identification, there are $(l_{max} - l_{min} + 1)$ separate runs required. Each row of Table 1 corresponds to such a separate run of the algorithms. But, a sorted collection C^q of elements for length q, is also sorted for length $(q - 1)$, whose values correspond to the $2(q - 1)$ MSBs in the collection. Thus, dropping the two LSBs of the elements of C^q would give the sorted collection $C^{(q-1)}$, for length $(q-1)$. Note that for Algorithm *CaTScan1*, as C^q holds elements that are representatives of forward and reverse complement of q-mers in S, an additional modification is necessary. While dropping the two LSBs of forward elements of C^q would correspond to forward elements of $C^{(q-1)}$, such an operation on the reverse complement elements of C^q would correspond to reverse complement elements of $C^{(q-1)}$ starting at one position to the right of the respective q-mers. Therefore, for all the elements in $C^{(q-1)}$ whose orientation bit is 1, their positional information is incremented by 1. The only two elements that will be missing in $C^{(q-1)}$ are $e^f_{(n-(q-1)+1)}$, the forward element of the last $(q-1)$-mer in S and e^{rc}_1, reverse complement element of the first $(q-1)$-mer in S. The positions of these two elements can be found using a binary search on $C^{(q-1)}$. Therefore, once the collection $C^{l_{max}}$ is sorted and the triplet l_{max}-mers identified, the patterns of lower lengths in the range $[l_{min}, l_{max}]$ could be identified by generating $C^{l_{max}} \rightarrow C^{(l_{max}-1)} \rightarrow C^{(l_{max}-2)} ... C^{l_{min}}$, saving the several creation and sorting steps of collection C required in the case of multiple runs. For Algorithm *CaTScan2*, the partition step is performed once to form A. Let $A^{l_{max}}_j$, $1 \leq j \leq 4^k$, be the array of partitions whose elements are sorted with respect to l_{max}-mers in S. After scanning $A^{l_{max}}$, to identify triplet l_{max}-mers, $A^{(l_{max}-1)}$ is obtained by

Table 1. Performance of *CaTScan1* & *CaTScan2* ($d = 1000$)

Genome	Size (Mbp)	Length (l)	*CaTScan1* Time(sec)	*CaTScan2* Time(sec)
C. elegans	100	20	475.09	623.25
		21	495.27	663.32
		22	527.24	651.28
		23	542.22	797.29
		24	559.00	893.83
D. melanogaster	118	20	586.62	697.36
		21	637.45	832.30
		22	627.76	779.54
		23	708.94	950.96
		24	688.99	1002.00

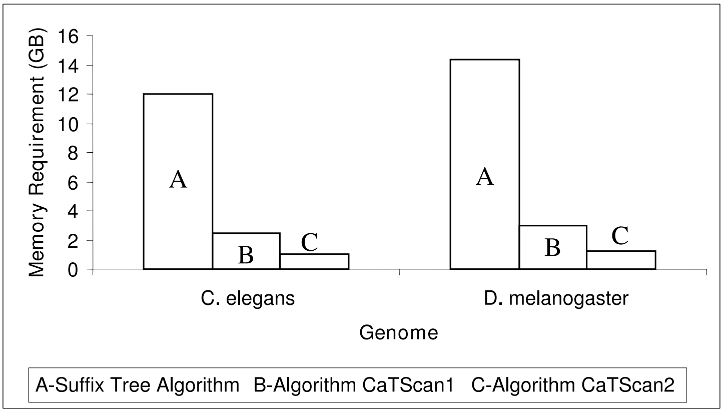


Fig. 2. Comparison of Memory Requirements

incrementing by 1, the positional information of elements in the partitions whose orientation bit is 1, and inserting $(1, 1)$ and $((n - (q - 1) + 1), 0)$, the positional and the orientational information of the reverse complement value of the first and the forward value of the last $(q - 1)$-mer in S, using a binary search in the partition corresponding to their $2k$-bit MSBs.

The runtime of algorithm *CaTScan1* to identify triplets of lengths in a specified range $[l_{min}, l_{max}]$ through separate runs for each length is $O(nl_{max}/w + n(l_{max} - 1)/w + ... + nl_{min}/w) = O((l_{max} + (l_{max} - 1) + ... + l_{min})n/w)$. By the above mentioned speedup strategy, the runtime will be $O(nl_{max}/w) + [(l_{max} - l_{min})(n + logn)]$. Similarly, the runtime of algorithm *CaTScan2* to identify triplets of lengths in a specified range $[l_{min}, l_{max}]$ through separate runs for each length is $O([l_{max} - l_{min} + 1]4^k + n[l_{max} - k] + n[(l_{max} - 1) - k] + ... + n[l_{min} - k]) = O([l_{max} - l_{min} + 1]4^k + n[l_{max} + (l_{max} - 1) + ... + l_{min} - (l_{max} - l_{min} + 1)k])$.

With the speedup strategy, the runtime will be $O([l_{max} - l_{min} + 1]4^k + n(l_{max} - k) + [(l_{max} - l_{min})(n + logn)])$. We are currently incorporating these speedup techniques in our programs.

We consider the reduction in memory required to process a given genome data to be the vital contribution of our work. Figure 2 shows a comparison of memory requirement of our algorithms with that of the Suffix tree algorithm (the memory requirement of the Suffix Tree Algorithm for the *D. melanogaster* genome is not available, hence a scaled value is shown). We believe that the reduction in memory requirement by a factor of 12 would make the processing of higher order genomes possible for detecting such endogenous controls and targets for RNAi by degradation. The estimated memory requirement of our algorithm for the human genome is approx. 30 GB. We are currently researching on developing a parallel algorithm that could identify such patterns of interest in higher order genomes.

5 Conclusion

In this paper we have discussed the Triple Repeat Identification Problem that has application in endogenous RNAi control and target discovery in genome sequences and proposed space and time efficient algorithms for the same. We have shown that one of our algorithms, when run on the *C. elegans* genome sequence, is 23 times faster and uses 1/12-th the memory compared to the known suffix tree approach of [7]. We hope that when we complete the implementation of our parallel algorithm, it would enable scanning genomes of higher order to discover such endogenous patterns of interest.

Acknowledgment

This research was supported in part by the NSF Grant ITR-0326155. We thank Ion I. Mandoiu for introducing the problem to us.

References

1. Agrawal, N., Dasaradhi, P.V.N.,Mohmmed, A., Malhotra, P., Bhatnagar, R.K., Mukherjee, S.K.: RNA Interference: Biology, Mechanism, and Applications. Microbiology and Molecular Biology Reviews (2003) 657-685
2. Buhler, J. and Tompa, M.: Finding motifs using random projections. Proc. Fifth Annual International Conference on Computational Molecular Biology (RECOMB) (2001)
3. Caplen, N.J., Mousses, S.: Short Interfering RNA (siRNA)-Mediated RNA Interference (RNAi) in Human Cells. Ann. N. Y. Acad. Sci. **1002** (2003) 56–62
4. Chalk, A.M., Wahlestedt, C., Sonnhammer, E.L.L.: Improved and automated prediction of effective siRNA. Biochemical and Biophysical Research Communications **319** (2004) 264-274

5. Fire, A., Xu, S., Montgomery, M.K., Kostas, S.A., Driver, S.E., Mello, C.C.: Potent and specific genetic interference by double-stranded RNA in Caenorhabditis elegans. Nature **391** (1998) 806–811
6. Gusfield, D.: Algorithms on Strings, Trees and Sequences. Cambridge University Press (1997)
7. Horesh, Y., Amir, A., Michaeli, S., Unger, R.: A rapid method for detection of putative RNAi target genes in genomic data. Bioinformatics **19(2)** (2003) ii73–ii80
8. Lim, L.P., Lau, N.C., Weinstein, E.G., Abdelhakim, A., Yekta, S., Rhoades, M.W., Burge, C.B., Bartel, D.P.: The microRNAs of Caenorhabditis elegans. Genes and Development **17** (2003) 991-1008
9. Rajasekaran, S., Balla, S., Huang, C.-H., Thapar, V., Gryk, M., Maciejewski, M., Schiller, M.: High-performance Exact Algorithms for Motif Search. Journal of Clinical Monitoring and Computing (Springer) **19(4-5)** (2005) pp 319–328
10. Rajasekaran, S., Balla, S., Huang, C.-H.: Exact Algorithms for Planted Motif Problems. Journal of Computational Biology **12(8)** (2005) 1117–1128

A Fast Approximate Covariance-Model-Based Database Search Method for Non-coding RNA

Scott F. Smith

Boise State University, Dept. of Electrical and Computer Engineering,
1910 University Ave., Boise, Idaho 83725-2075, USA
SFSmith@BoiseState.edu

Abstract. A covariance-model-based search method for non-coding RNA genes is proposed which is much faster than dynamic programming, but which is shown to be very effective in experimental tests. The method incorporates secondary structure information in the entire first pass of the database, unlike the usual primary-sequence-only pre-filters applied when using dynamic programming. An iterative alignment refining algorithm which starts at an ungapped alignment and successively selects alignment breakpoints gives only an approximation to the optimal alignment, but appears to be sufficient for gene localization.

Keywords: Non-coding RNA, functional RNA, covariance models, database search, gene finding, dynamic programming.

1 Introduction

Non-coding RNA (ncRNA) are biological macromolecules that are transcribed from DNA, but do not need to be further translated into protein to accomplish their biological function. They are single-stranded and form their functional three-dimensional shape as a result of which sequence positions are intra-molecularly base paired with which other positions. Normally, similar base pairing patterns (secondary structures) are sufficient to identify two ncRNA molecules as having similar function, although primary sequence similarity may add additional information to identify homologs. Models using primary sequence alone are usually not powerful enough for ncRNA search.

The annotation of genome sequences with gene locations is done much differently depending on whether the functional molecule of the gene is protein or RNA. Typically, protein-coding genes are determined in a two-step process, where putative gene locations are first found without regard to specific protein family [1] and then these putative genes are classified by family [2, 3]. Such a two-step process is not currently feasible for non-coding RNA (also called functional RNA) because no known statistical method is able to adequately classify chromosomal sequence regions into those with generic non-coding RNA genes versus those without generic ncRNA genes [4]. Gene finding programs for ncRNA therefore depend on searching sequence databases using a model of a particular ncRNA family.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 270–281, 2007.

One of the most successful ncRNA gene search models is the covariance model (CM) [5, 6]. The name derives from the inclusion of secondary structure information in the model, in which the individual nucleotides are allowed to vary considerably as long as base-paired nucleotides co-vary in a manor that maintains the likelihood of base pairing. The major drawback of using a CM for database search with optimal dynamic-programming methods is that the computational resources required are enormous. The solution to this problem is normally to resort to primary-sequence-based pre-filters such as BLAST [7] or HMMs to greatly reduce the amount of database presented to the CM search algorithm. The problem with using BLAST is that there is no guarantee that sufficient primary-sequence homology exists such that true ncRNA gene locations are not discarded. Using the methodology of Weinberg and Ruzzo [8], HMM parameters and score thresholds can be determined that guarantee no database regions are discarded which the CM algorithm will subsequently label as a gene, but the computational requirements of this method are still quite large.

An alternative approach is presented here, where the secondary structure information in the CM is used right from the initial scan of the database. An approximate refinement method for the alignment of the database to the sequence and structure of the model is used to quickly find potential score improvements. It should be pointed out that the resulting alignments are only useful for determining high-scoring portions of genomic data and that full dynamic programming alignment should be undertaken on the high-scoring locations if the alignment itself is of interest for further study. The search begins from the observation that there is often at least some significantly large range within the database sequence which is ungapped with respect to the consensus sequence of the CM and that this ungapped range may contain entire base-paired structures. The database search is initially an ungapped scan of the database using the covariance model with insertion and deletion states removed. A very loose score threshold is then applied and alignment improvement is done on those database positions with scores exceeding the loose threshold. The refinement method involves iteratively determining breakpoints in the model and searching over insertions and deletions only at the current breakpoint. Once no improvement is found at the current breakpoint, the process is halted.

In experimental investigations of ncRNA genes in the *C. elegans* [9] complete genome, it is found that an ungapped scan alone sometimes turns up false positives as defined by the Rfam [10] ncRNA families database. However, the alignment improvement method cleanly partitions true family members from false positives. It is also noted that identification of false positives is made easier in that false positives with high initial ungapped scan scores typically show no improvement during refinement, whereas true family members often improve.

A quick overview of using CMs for database search is given in Section 2. The search method being proposed is detailed in Section 3. Experimental results for a number of ncRNA families are given in Section 4 using the *C. elegans* genome as database. Section 5 gives some concluding comments.

2 Covariance Models for Non-coding RNA Database Search

A covariance model is estimated from a group of nucleotide sequences which have been determined to form a family of ncRNA genes. This model may then be used to search nucleotide databases for new instances of the family or to align a sequence with the family model for further study. The task of interest here is the former, where all that is required is for the method to return locations within the nucleotide database that hold instances of the ncRNA family with a high level of confidence. In this work the CM structure and parameter values will be taken as given and the object is to efficiently determine the database locations using the given CM for a family.

The basis for the CM structure is a secondary-structure-annotated consensus sequence for the family. The ncRNA molecule is single-stranded with intra-molecular base paring. Each position in the consensus sequence is labeled as an unpaired or a base-paired position. The base-paired positions are further labeled as having the other position of its pair either to the right (3'-direction) or left (5'-direction). The CM structure does not allow pseudoknots, so this secondary-structure labeling is unambiguous. If the true structure does in fact contain a pseudoknot, then some of the base-paired positions must be labeled as if they were unpaired, which results is some loss of information in the model and some loss of database search power.

2.1 From Structure-Annotated Consensus Sequence to a Covariance Model Structure Tree

Each unpaired consensus sequence position is assigned to a node in the CM structure and each pair of base-paired positions is assigned one node. If there were no base pairs, then the nodes could be arranged in a single chain mirroring the consensus sequence order and the model would degenerate into a profile hidden Markov model. In order to allow base pairs, a base-pair node (P node) builds on the structure between the two base-pair positions, unlike in a profile HMM where each node always builds on the structure to its left. To build unpaired structure around a P node, the CM needs both L nodes and R nodes which build one position to the left and right respectively. The profile HMM can therefore be viewed as a CM containing only R nodes. Finally, to join sections of model containing non-nested pairs, a bifurcation node (B node) is required. As with a profile HMM the CM also contains start nodes (S nodes) and end nodes (E nodes), although the functions of these nodes is reversed as a CM is evaluated from E nodes toward S nodes (the opposite of an HMM).

Figure 1 shows an example annotated consensus sequence and the CM node tree it generates. The dash symbol indicates a position labeled as not base paired, a < symbol indicates the left-hand position of a pair, and the > symbol is the right-hand position. The B node is needed since a stem of three base pairs on the left is not nested within the stem of two base pairs on the right (or vice versa). Each branch has an S node at the top and either a B node or an E node at the bottom. The consensus sequence can be read from the node tree by starting at the top (the root start node) and reading counter-clockwise around the tree. Whenever there is a choice about using an L or R node for an unpaired position, the L node is used such that CM trees are unique given a consensus secondary structure.

The nucleotide labels (A, C, G, or U) on the tree represent only the most probable nucleotide at the given position. The CM has parameters representing the probabilities of each nucleotide (or nucleotide pair) emitted by each node. For the single-emission L and R nodes, there are four probabilities and for P nodes, there are sixteen probabilities. These probabilities are normally given as log likelihood ratios where base-2 logarithms are used. The resulting model scores can then be calculated by addition (using the assumption of independence) and an increase of one unit in the score may be interpreted as an increase in probability by a factor of two.

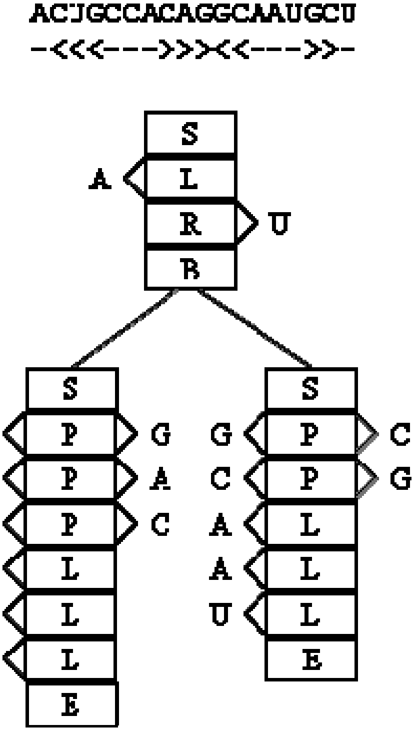


Fig. 1. Structure-Annotated Consensus Sequence and Associated CM Node Tree

So far, the model described is similar to an ungapped position-specific scoring matrix. The main difference is that the scoring for paired nodes is taken simultaneously. A full CM also includes internal state structure for each model node which allows the deletion of the consensus position or the insertion of an arbitrary number of database symbols between consensus positions. This is similar to the inclusion of deletion and insertion states at each node of a profile HMM in addition to the consensus match state of each node. This internal state structure is somewhat more complicated in a CM, particularly in the P nodes where either the left or the right or both consensus positions may be deleted. Position-specific insertion initiation, insertion continuation, deletion initiation, and deletion continuation penalties are implemented as in the profile HMM through the transition scores between the states. All of the emission and transition score parameters are estimated from the secondary-structure-annotated multiple alignment of the member sequences of the ncRNA family. Pseudocounts are normally included to avoid any log likelihood scores of minus infinity.

2.2 Searching a Database with a Covariance Model

The traditional method of using a CM to search a database is to employ dynamic programming. Scoring starts at the E states (in the E nodes) and progresses toward the root S state (in the root S node). The score is taken as 0 at the E states with each E state representing a null sequence. Scores are calculated at a state for each possible position in the database and each possible number of database symbols leading up to that position. To make the calculation feasible, a maximum length cutoff is used for the state score calculations. This cutoff must be at least as long as the longest new family member that is expected to be found in the database (a number that is hard to know in advance). The maximum score taken over all lengths for a given database position at the root start state is the overall score of the model at that database position. It should be clear that these calculations require large amounts of computing resources. Some ncRNA families with long consensus sequences can take years of processing on a single modern processor to cover a giga-base of data [8]. Since there are currently more than five hundred Rfam families and sequencing projects underway are expected to produce many more giga-bases of genomic data [11], this situation is clearly unacceptable.

The solution used to generate Rfam is to use BLAST with a very loose threshold to discard portions of the database that do not have much primary-sequence homology at all with the CM consensus sequence. The use of the secondary structure information in the CM is limited to sequences that have enough primary-sequence homology to pass the initial scan. Since functional RNA molecules have much more conserved secondary structure than primary sequence, a major advantage of using the CM in the first place can be lost. It is not clear exactly how loose the threshold should be. If chosen to high, true family members will not be found. If chosen too low, the amount of database discarded will be small and computation times long. The alternative method of Weinberg and Ruzzo [8] eliminates this uncertainty, but HMM searching is much slower than BLAST and the amount of database not discarded can still be large enough for the dynamic-programming CM search to still be slow.

3 Fast Search Method

If one takes at look at the family multiple alignments in Rfam, it is apparent that there is almost always a range of nucleotides in every family member that forms an ungapped alignment with the consensus sequence and that these ranges often contain both halves of a stem (nested base pairs). This implies that there should be some signal present in the scores of ungapped local alignments of the consensus sequence of a family with the database. To the extent that there are no gaps between base pairs, the sixteen-valued pair-wise match scores should add even more to the signal produced. Those portions of the alignment that do not match due to gaps will contribute noise. The net result should be a tendency for true ncRNA family genes to generate higher scores with ungapped alignment than random sequences.

As shown in the experimental section that follows, the signal-to-noise ratio of the ungapped scan alone is not enough to exclude false positives. However, an

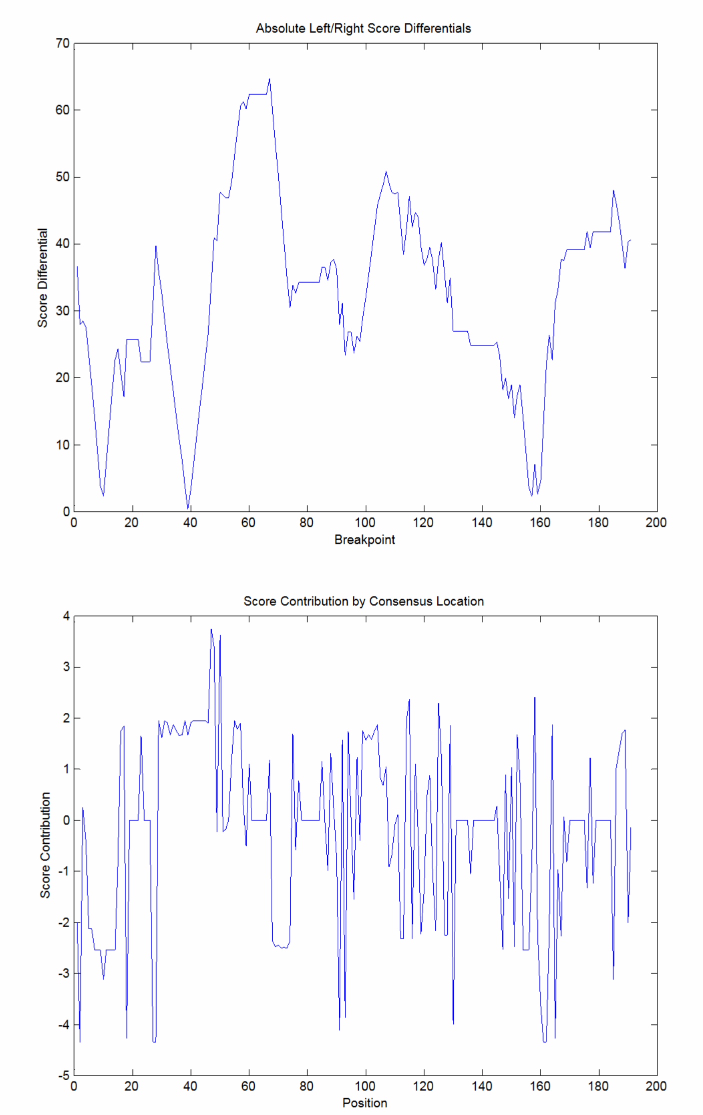


Fig. 2. Absolute Score Differentials by Breakpoint and Score Contributions by Position for a U2 Gene on Chromosome I (forward strand) at Start of First Round of Alignment Improvement

approximate alignment improvement procedure that iteratively chooses breakpoints in the consensus sequence and tries a range of insertion and deletion lengths at the breakpoint appears to increase the signal-to-noise ratio enough to be useful. In particular, the breakpoint in a somewhat-higher-than-average-scoring ungapped alignment is chosen by finding the consensus sequence location that maximizes the absolute difference in partial scores to its left versus its right. The breakpoint is therefore the position that maximizes

$$\mathrm{abs}(\Sigma_i\, l_i - \Sigma_j\, r_j), \qquad (1)$$

where l_i is a score contribution of the alignment of consensus position i to the left of the breakpoint and r_j is a similar contribution for a position j to the right. If the left sum is greater than the right sum at the breakpoint, the alignment to the left should be retained and the portion to the right realigned (and vice versa). This realignment takes of the form of a shift of the entire consensus sequence to one side of the breakpoint. If this refinement of the alignment increases the score, a new breakpoint is chosen and the procedure repeated until no improvement is seen.

Figure 2 shows the first round of this alignment improvement scheme for a true member of the U2 ncRNA family on the forward strand of chromosome I in *C elegans*. The top plot shows the absolute value of the score differential as given by Equation 1. The maximum value at consensus sequence position 67 shows that the breakpoint will be chosen there. It is not possible to tell from the figure, but the sum to the left of position 67 is greater than the sum to the right (by about 65). A series of possible insertions or deletions will be tried at position 67 (up to ten insertion and up to ten deletions are used in the results section below). The bottom plot in Figure 2 shows the score contributions at each consensus location. It can be seen that there is a well-aligned segment between positions 30 and 67. Since the poorly aligned segment to the right is much larger, it contributes more to the left/right score mismatch and is attacked first. If the score improves by realigning to the right of 67, then the portion to the left of 30 is likely to be realigned in a later round.

4 Experimental Results

The scores generated by the proposed database search method for several ncRNA families applied to the complete *C. elegans* genome are analyzed. The U2 family is investigated in detail and the results for the other families summarized. The families were chosen because they are relatively abundant in *C. elegans* (7 - 60 genes). However, the most abundant ncRNA gene (for tRNA) was not run because it has so many instances (as many as 279 per chromosome) that it is hard to analyze.

Table 1 shows the eighteen members of the U2 family (part of the major spliceosome). There are six chromosomes (I, II, III, IV, V, and X) and the U2 gene can be either on the forward strand or the reverse complement strand, resulting in twelve database sequences to process. The chromosome sequences are those of WormBase Release WS160 and the ncRNA families and covariance models all come from Rfam 7.0. The length of the six chromosomes are 15.1, 15.3, 13.8, 17.5, 20.9, and 17.7 Mbp, for a total of 100.3 Mbp. Since both forward and reverse complement are needed, a total of about 200 million bases are processed.

The results of the processing are presented in Table 2, both in per-chromosome and whole genome form. If the ungapped score exceeded -70, then the alignment at the location was refined using the breakpoint method described in Section 3. The scores of the true U2 genes (as defined by Rfam) are shown both before (Init) and after (Imp) the alignment improvement scheme. No true genes were missed since the lowest first-pass score was -44.59 and the cutoff was -70. In all cases, the breakpoint method generated a higher score than the original ungapped scan.

Table 1. The Eighteen U2 ncRNA Genes (Rfam RF00004) in *C. elegans*

Chromosome	EMBL Clone / Clone Start	Chromosome Start
I (BX284601)	Z93391.1/20231, Z81504.2/2509, Z81553.1/6257, Z81553.1/7099	11657554, 12153034, 12293797, 12296458
I Reverse Complement	AL031636.1/4082, Z981504.2/7291, Z81105.2/12802, Z81553.1/8918, AL009246.1/8935	11636405, 12157816, 12181962, 12296458, 12332995
II (BX284602)*	Z81495.1/26606, Z82076.1/18363	13852675, 13951308
II Reverse Complement	AL032648.1/23803, Z81495.1/3313, Z81495.1/9940, Z82076.1/16099	13717643, 13829382, 13836009, 13949044
III	None on either strand	
IV R.C. (BX284604)	AF045635.1/5423, Z99281.1/127762	624275, 14765001
V R.C. (BX284605)	Z92830.1/35793	16228471
X	None on either strand	

***Rfam lists a third U2 gene on the forward strand of Chr. II at 13944168. However, this gene is not listed in EMBL (as all the others are) and it is not found by the search program (as all others are).**

The highest-scoring false positive U2 gene is also shown in Table 2 for each chromosome. The location generating the initial highest score and that generating the highest score after alignment improvement may or may not be the same. The number of false positives over the -70 initial score threshold which underwent the score improvement process is given. It turns out that very few of the false positives actually increased in score as a result of the alignment improvement procedure. This is hinted at by the fact that the initial and improved highest false scores are often the same.

The final column of the table lists the score margin. This is the difference between the lowest scoring true positive and the highest scoring false positive. This margin is seen to increase in all cases implying that the separation between true and false positives have become better. In the overall results at the bottom of the table, it can be noted that the margin is negative before refinement. The negative margin means that using a score threshold that gets all true positives will also accept at least one false positive. In fact, it is clear from the table that at least three false positives would be accepted (with scores -37.16, -39.56, and -44.36). The use of ungapped scanning alone is not sufficient to separate true from false.

It should be pointed out that the scores here have no meaning in an absolute sense because state transition values have not been used. An ungapped parse of a sequence to the covariance model will depend on the emission scores plus a constant equal to the sum of the transition scores from one match state to the next. This constant can be absorbed into the score thresholds. Differences in scores (such as margins) are unaffected.

Table 2. Scores of U2 in *C. elegans* Using Fast CM Search Method

Xsome	Initial and Improved Scores (True Genes)	Highest False Gene	Number False	Margin
I	Init: -41.89, -40.01, -40.63, -40.63 Imp: -24.48, -18.30, -22.64, -22.64	Init: -51.95 Imp: -51.95	25	10.06 27.47
I R.C.	Init: -39.57, -40.01, -40.63, -40.63, -40.63 Imp: -22.15, -18.30, -22.63, -23.22, -22.64	Init: -53.34 Imp: -40.32	40	12.71 17.10
II	Init: -40.63, -39.63 Imp: -22.63, -22.63	Init: -49.01 Imp: -44.52	26	8.38 21.89
II R.C.	Init: -40.09, -40.63, -40.63, -39.63 Imp: -22.63, -22.63, -22.63, -22.63	Init: -53.41 Imp: -53.41	29	12.78 30.78
III	None	Init: -51.79 Imp: -51.79	22	N/A
III R.C.	None	Init: -46.44 Imp: -46.44	31	N/A
IV	None	Init: -44.36 Imp: -36.32	39	N/A
IV R.C.	Init: -44.59, -36.74 Imp: -21.39, -18.47	Init: -55.97 Imp: -55.97	28	11.38 34.58
V	None	Init: -51.87 Imp: -51.61	31	N/A
V R.C.	Init: -43.06 Imp: -7.41	Init: -53.27 Imp: -53.27	38	10.21 45.86
X	None	Init: -39.56 Imp: -39.56	30	N/A
X R.C.	None	Init: -37.16 Imp: -37.16	19	N/A
Overall	Init: -44.59 (Chr. IV R.C. - AF045635.1) Imp: -24.48 (Chr. I - Z93391.1)	Init: -37.16 Imp: -37.16	328	-7.43 12.68

False genes are those positions generating a score greater than -70 and which are not identified in Rfam as a U2 gene.

The results for several other families (U5, Histone3, 5S_rRNA, and K_chan_RES) along with the overall results from U2 (Table 2) are summarized in Table 3. The number of positions in the consensus sequence of the CM is shown along with the total number of true genes from Rfam on all six chromosomes of *C. elegans* (both forward and reverse strands). The lowest score for a true ncRNA gene is shown both before and after alignment improvement, where the before and after scores are not necessarily for the same gene. The highest false score found anywhere in the genome is reported both before and after alignment improvement as well as the number of false genes for which a attempt at alignment improvement was made. The score threshold used to determine if improvement was to be attempted at a position is shown, which is always chosen well below the lowest initial true score and low enough to capture several hundred false genes. Finally, the margin is calculated as the difference between the lowest true score and the highest false score for both initial scan and after improvement.

In all cases, the alignment improvement operation did not lower the margin. Since this operation used much less computation time than the initial scan, there is little reason not to include it. The summary results show no change between initial and improved scores for 5S_rRNA and K_chan_RES, however, some of the scores other than the lowest true and highest false did in fact change in these cases. In the Histone3 case, very few true scores actually changed, with the lowest true score being one of the exceptions. A look at the multiple alignments for Histone3 family members (across all species, not just *C. elegans*) shows very little insertion or deletion behavior, so there is very little room to improve alignments with respect to the initial ungapped scan, since the true alignment is usually already ungapped.

Table 3. Scores of ncRNA Family Genes in *C. elegans* Using Fast CM Search Method Over Whole Genome

ncRNA Family	Consensus Length	Numb. in *C. elegans*	Lowest True Score	Highest False Score	Num. False	Thresh.	Margin
U5	117	12	Init: 17.31 Imp: 31.39	Init: 19.38 Imp: 26.70	493	0	Init: 2.07 Imp: 4.69
Histone3	26	60	Init: 25.75 Imp: 28.07	Init: 25.78 Imp: 25.78	461	15	Init: -0.03 Imp: 2.29
5S_ rRNA	119	13	Init: -0.09 Imp: -0.09	Init: -36.66 Imp: -36.66	439	-70	Init: 36.57 Imp: 36.57
K_chan _RES	114	7	Init: 25.81 Imp: 25.81	Init: 11.17 Imp: 11.17	932	-50	Init: 14.64 Imp: 14.64
U2	191	18	Init: -44.59 Imp: -24.48	Init: -37.16 Imp: -37.16	328	-70	Init: -7.43 Imp: 12.68

It is not entirely straightforward to determine from Rfam the complete set of ncRNA genes of a particular family in an organism. Using the "genomes" tab on the Sanger Institute Rfam mirror site [12] leads to tables of all ncRNA genes in each family by *C. elegans* chromosome. Forward strand versus reverse strand genes can be identified by whether the location range is increasing or decreasing in the tables. If one instead goes to the full multiple alignments for a particular family and selects only those sequences labeled "Cae.ele.", a different set of genes is often found. These two sets of genes are normally almost, but often not quite, identical. In cases where the two lists differed, other sources such as EMBL [11] were consulted for ncRNA gene annotation. Table 4 is supplied with the actual numbers of genes taken as true genes on the forward and reverse strand of each chromosome for each family to aid in replication of results.

Table 4. Number of ncRNA Family Genes by Chromosome in *C. elegans* (forward strand on left of slash / reverse strand on right)

ncRNA Family	I	II	III	IV	V	X
U5	1/0	0/0	0/0	7/4	0/0	0/0
Histone3	1/1	6/7	0/0	11/10	11/11	1/1
5S_rRNA	0/0	0/0	0/0	0/0	11/2	0/0
K_chan_RES	0/0	1/2	0/0	0/1	1/1	1/0
U2	4/5	2/4	0/0	0/2	0/1	0/0

The CPU time used to run the initial ungapped scan of the 15.1 Mbp chromosome I forward strand for the U2 family was 393 seconds on a 2.8MHz Pentium IV when written in MATLAB. The alignment improvement time was much less than one second for 346 improvement attempts. Initial scan times for the U5, Histone3, 5S_rRNA, and K_chan_RES models on forward Chr. I were 340, 264, 357, and 346 seconds respectively. A program that did nothing except read in the database file and convert the database symbols A, C, G, and T into index numbers 1, 2, 3, and 4 used to access model score values took 238 seconds for Chr. I, so a large amount of the time is overhead not involved in the actual scoring calculation. These times would be expected to improve significantly if the program was converted to C and optimized for speed.

Weinberg and Ruzzo [8] list run time results for the U5 covariance model using the same processor (2.8MHz Pentium IV). With the HMM lossless pre-filter followed by standard dynamic-programming CM search, a run time of 8.9 days was required to process a 8 giga-base database. Using dynamic-programming CM search on the entire 8 giga-base database, an estimated 1081 days would be required. The fast method presented here is predicted to take 340*8000/15.1 = 180,132 seconds, or 2.1 days. The factor of more than four improvement in speed in likely a gross underestimate on what can be accomplished using more efficient coding.

5 Conclusion

A fast method for ncRNA gene search using covariance model parameter files has been presented. On a limited test set of five ncRNA gene families tested against the *C. elegans* genome, the method appears to estimate the score of the optimal dynamic programming CM search method well enough to return essentially the same results with significantly less use of computational resources.

The current form of the code used to generate experimental results needs to be significantly upgraded before large-scale testing can take place. Some hand conversion of structural information in the covariance model parameter files is still required. This needs to be automated before testing of all know family models (over 500 to date) can be tried against a given genome. Also conversion from MATLAB to a more efficient language needs to take place before processing larger genomes such as that of human.

It has also been noted by the author that there is position-specific information about the likelihood of insertions or deletions in the CM parameter file that could be combined with the breakpoint estimator used here to potentially improve breakpoint estimation. Finally, more investigation of the best choice of score threshold for passing database positions from the initial ungapped scan on to the alignment improvement stage need to be undertaken.

Acknowledgments. The project described was supported in part by NIH Grant Number P20 RR016454 from the INBRE Program of the National Center for Research Resources.

References

1. Burge, C., Karlin, S.: Prediction of Complete Gene Structures in Human Genomic DNA. J. Mol. Biol. 268 (1997) 78–94
2. Eddy, S.: Profile Hidden Markov Models. Bioinformatics 14 (1998) 755–763
3. Finn, R., Mistry, J., Schuster-Böckler, B., Griffiths-Jones, S., Hollich, V., Lassmann, T., Moxon, S., Marshall, M., Khanna, A., Durbin, R., Eddy, S., Sonnhammer, E., Bateman, A.: Pfam: Clans, Web Tools and Services. Nucleic Acids Res. 34 (2006) D247–D251
4. Rivas, E., Eddy, S.: Secondary Structure Alone is Generally not Statistically Significant for Detection of Noncoding RNAs. Bioinformatics 16 (2000) 583–605
5. Eddy, S., Durbin, R.: RNA Sequence Analysis Using Covariance Models. Nucleic Acids Res. 22 (1994) 2079–2088
6. Durbin, R., Eddy, S., Krogh, A., Mitchison, G.: Biological Sequence Analysis. Cambridge University Press, Cambridge UK (1998)
7. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic Local Alignment Search Tool. J. Mol. Biol. 215 (1990) 403–410
8. Weinberg, Z., Ruzzo, W.: Faster Genome Annotation of Non-coding RNA Families Without Loss of Accuracy. Int. Conf. Res. Comp. Mol. Bio. (2004) 243–251
9. WormBase: http://www.wormbase.org
10. 10. Griffiths-Jones, S., Moxon, S., Marshall, M., Khanna, A., Eddy, S., Bateman, A.: Rfam: Annotating Non-coding RNAs in Complete Genomes. Nucleic Acids Res. 33 (2005) D121–D124
11. Ensembl: http://www.ensembl.org
12. Rfam: http://www.sanger.ac.uk/Software/Rfam/

Extensions of Naive Bayes and Their Applications to Bioinformatics

Raja Loganantharaj

Bioinformatics Research Lab, University of Louisiana, Lafayette LA 70504
logan@cacs.louisiana.edu

Abstract. In this paper we will study the naïve Bayes, one of the popular machine learning algorithms, and improve its accuracy without seriously affecting its computational efficiency. Naïve Bayes assumes positional independence, which makes the computation of the joint probability value easier at the expense of the accuracy or the underlying reality. In addition, the prior probabilities of positive and negative instances are computed from the training instances, which often do not accurately reflect the real prior probabilities. In this paper we address these two issues. We have developed algorithms that automatically perturb the computed prior probabilities and search around the neighborhood to maximize a given objective function. To improve the prediction accuracy we introduce limited dependency on the underlying pattern. We have demonstrated the importance of these extensions by applying them to solve the problem in discriminating a TATA box from putative TATA boxes found in promoter regions of plant genome. The best prediction accuracy of a naïve Bayes with 10 fold cross validation was 69% while the second extension gave the prediction accuracy of 79% which is better than the best solution from an artificial neural network prediction.

Keywords: machine learning, computational efficiency, improved performance.

1 Introduction

Naïve Bayes has been successfully used in Bioinformatics for classifying and predicting [1-3] and as well as for fusing information [4]. Naïve Bayes is computationally efficient and practically very useful since it assumes that the output is dependent only on the current state (Markov assumption). The convenience and the efficiency come by sacrificing the underlying reality. In addition, the prior probabilities of classes are computed from the training set in many application programs including the popular data mining tool WEKA [5]. Such computation does not accurately reflect the real prior probabilities since the training data usually does not have a representative mixture of positive and negative instances so as to reflect reality. In this paper we address those two issues.

Obtaining prior probability of an event or an object is a non-trivial task and statisticians have studied the problem in details [6]. The prior probability may be assigned using one of the following methods: non-informative prior, subjective-expert judgment prior, and Monte Carlo priors. Each of these methods has their pros and

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 282–292, 2007.

cons and Monte Carlo priors seem to be unbiased, but it require considerable amount of computation for the simulation. We take a data-dependent approach that automatically perturb the computed prior probabilities and search around the neighborhood to maximize a given expected utility. We provide the details of this approach in section 2.

In this paper we investigate relaxing Markov order 0 assumption of naïve Bayes to improve the prediction accuracy without increasing the computational cost. By making the outcome dependant on the partial pattern, i.e. Markov order 1 to r, we can improve the prediction accuracy.

We have demonstrated these extensions to solve the problem of discriminating a TATA box from putative TATA boxes found in promoter regions of plant genome.

This paper is organized as follows. A formulation of the problem is described in section 2 along with the proposed extensions one and two for the naïve Bayes to improve its prediction ability. In section 3 we introduce the application domain and apply these extensions to solve the problem and show the results. In section 4, we summarize the contribution and discuss the results.

2 Formulation

A learning algorithm creates decision boundaries among the classes of the labeled training instances so that it can classify an unlabelled instance correctly. Suppose there are *r* different classes in the training set, and each labeled training instance has *k* features. Once a Bayesian net is trained with the training set, it computes the probability of each class for a given instance and classifies the unknown instance to a class that has the highest probability given the features of the instance. If $P(C_m| e_1,e_2,..,e_k)$ represents the probability of the class C_m for the given set of features $e_1,e_2,..,e_k$, the Bayesian network classify the instance to be belonging to C_n if $P(C_n| e_1,e_2,..,e_k)$ is the maximum among $P(C_m| e_1,e_2,..,e_k)$ for all *m* from 1 to *r*.

$$P(C_m| e_1,e_2,..,e_k)$$

$$= P(C_m , e_1,e_2,..,e_k)/P(e_1,e_2,..,e_k) \qquad \text{Using Bayesian theorem}$$

Using chain rule, we will get

$$= P(e_1|e_2,..,e_k,C_m).\ P(e_2|e_3,..,e_k,C_m)\ldots\ P(e_k| C_m).\ P(C_m)/P(e_1,e_2,..,e_k) \qquad 1$$

The computation is simplified by assuming positional independence, that is $P(e_n|e_{n+1},..,e_k,C_m) = P(e_n|C_m)$. Using positional independence (naïve Bayes assumption) $P(C_m| e_1,e_2,..,e_k)$ becomes

$$C.\ P(C_m).\prod P(e_n|C_m) \text{ for } n=1 \text{ to } k \text{ where } C = 1/P(e_1,e_2,..,e_k) \qquad 2$$

Notice that $P(e_n|C_m)$ and $P(C_m)$ are pre-computed from the training set for *n* from 1 to *k* and *m* from 1 to *r*.

2.1 First Improvement

The outcome of the naïve Bayes classifier is also influenced by the prior probabilities of the classes in the training set as illustrated in formula 2. Obtaining the correct prior probabilities of these classes is a non-trivial task. Naïve Bayes computes these prior probabilities from the instances of the training set which usually does not reflect the correct mixture of positive and negative instances. To solve this problem, we perturb the computed prior probabilities of these classes so as to maximize some desirable objective function of a prediction algorithm. One such objective function is to maximize either the prediction accuracy or the separation between true positive and false positive, or some combination of both. Suppose we consider a binary classifier which consists of two classes representing positive and negative instances in a training set. Let Tot_p and Tot_n respectively represent total positive and total negative instances in the training set. Suppose a predictive algorithm correctly predicts Pred_tp of positive instances and Pred_tn of negative instances. The following metrics are defined as follows:

Table 1. Definition of terms

True positive (TP)	=	Pred_tp/Tot_p
True negative (TN)	=	Pred_tn/Tot_n
False positive (FP)	=	1- Pred_tn/Tot_n
Prediction accuracy (PA)	=	(Pred_tp+Pred_tn)/(Tot_p+Tot_n)
Separation between TP -FP	=	TP+TN -1
Combined utility	=	λ PA + (1- λ)(TP – FP) where $\lambda \in [0..1]$

Since the computed value of prior probability is not accurate, it is reasonable to perturb the prior probability around the computed value so as to maximize the objective function. Researchers have used either prediction accuracy or the maximum separation between true positive and false positive as an objective function to measure the effectiveness and to compare different prediction algorithms. We will use the combined utility in Table 1 as our objective function with λ set to some fixed value, for example at 0.75. We will demonstrate this approach by applying this to a problem in bioinformatics.

2.2 Second Improvement

Let us recall the Bayesian formulation of classification

$$P(C_m| e_1,e_2,..,e_k)$$
$$= P(e_1|e_2,..,e_k,C_m).\ P(e_2|e_3,..,e_k,C_m)\ldots\ P(e_k|\ C_m).\ P(C_m\ )/P(e_1,e_2,..,e_k)$$

In naive Bayes, positional independence is used, that is $P(e_g|e_{g+1},..,e_k,C_m)$ is assumed to be equal to $P(e_g|C_m)$ which reduces the accuracy of the overall prediction algorithm. Since the probability is dependent only on the current class independent of the neighboring instances, is called Markov order 0 assumption. We are proposing an approach that improves the accuracy without increasing computational cost, but somewhat increased in space to maintain the probability table. We use limited dependency that can vary from 0 through k where dependency 0 is the naïve Bayes. We will show the corresponding formula for the limited dependency assumption.

$$P(C_m| e_1,e_2,..,e_k)$$
$$= P(e_1|e_2,..,e_k,C_m). P(e_2|e_3,..,e_k,C_m)\ldots P(e_k| C_m). P(C_m)/P(e_1,e_2,..,e_k)$$

With Markov order 1 it becomes

$$= P(e_1|e_2,C_m). P(e_2|e_3,C_m)\ldots P(e_{k-1}| e_k , C_m). P(e_k| C_m). P(C_m)/P(e_1,e_2,..,e_k)$$
$$= C. P(C_m). P(e_k| C_m)\prod P(e_n| e_{n+1},C_m) \text{ for } n=1 \text{ to } k-1 \text{ where } C = 1/P(e_1,e_2,..,e_k)$$

This can be generalized to Markov order *g*.

$$P(C_m| e_1,e_2,..,e_k)$$
$$= C. P(C_m). P(e_k| C_m). P(e_{k-1}| e_k\ C_m)\ldots P(e_{k-g+1}| e_{k-g+2},..\ e_k\ C_m)\prod P(e_n| e_{n+1}, e_{n+2,..,} e_{n+g}C_m)$$
$$\text{for } n=1 \text{ to } k\text{-}g \text{ where } C = 1/P(e_1,e_2,..,e_k)$$

Let us describe some implementation details. Suppose each element e_n takes d different values. To maintain *g* dependency, it will take table of size *d*(g+1)* compared to a table size *d* for naïve Bayes. We will illustrate the improved accuracy of this extension in the next section.

3 Application of These Extensions

To understand a regulatory mechanism, it is important to detect all the binding sites in a promoter region. Several approaches for finding binding sites in a promoter region have been proposed in literature and Tompa et al. [8] have recently studied several tools and found that no single tool performed well on all the data sets they have tested. In this work, our emphasis is to compare the proposed extensions with respect to naïve Bayes. A TATA box is a common transcription binding site occurs in upstream of a core promoter region [9]. It is usually detected by matching the profile 5'-TATAWAW-3' where W is either A or T. Unfortunately there are several substrings in the neighborhood of a TATA box fit to the profile which make the problem of recognizing the real TATA box from putative TATA boxes hard.

Our previous work [10, 11] on discriminating TATA box from putative TATA boxes has revealed that the neighborhood around a TATA box carries the information required to distinguish TATA box from putative TATA boxes.

Among many methods, a position specific weighted matrix (PSWM) has been used very successfully to detect a motif in a sequence knowing the profiles of the motif. We, therefore, use a PSWM to detect all the putative TATA boxes from a given set of promoter sequences. The problem of discriminating a TATA box from a set of putative TATA boxes becomes a problem of a binary classification. A machine learning algorithm can be applied to learn the patterns from the known set of TATA and non TATA boxes, and then be used to classify an unknown putative TATA box into a TATA or a non TATA box.

We have downloaded promoter sequences of plant genome from PlantProm DB (http://mendel.cs.rhul.ac.uk/mendel.php?topic=plantprom) [12], an annotated non-redundant collection of proximal promoter sequences for RNA polymerase II with experimentally determined transcription start site(s) (TSS) from various plant species. The current release of PlantProm DB contains 305 entries, of which 71 are monocot, 220 are dicot and 14 are other plants. Each promoter sequence is of length 250 bp from -200 to 50 with transcription start site (TSS) located at 1.

We started the experiment with the detection of all TATA boxes in TATA promoters using PSWM. The putative TATA box found in the promoters region between -40bp to -10bp from the TSS is indeed the active TATA box. We have collected the substrings of length 15bp in the downstream and in the upstream from the core TATA box (TATAWAW) and these strings form the positive instances. We have scanned the promoter regions from -200bp to -40bp for TATA box and the substrings flanking these putative TATA boxes become the negative instances. The probability of finding a TATA box by chance in a genomic sequence is $(1/4)^7$, that is, one can find a TATA box in every sequence of length 16,384 bp by chance. We found abundance of putative TATA boxes in the promoter region, about 2 or 3 within 160bp length. Since negative instances were well over 2 times of that of positive instances, we have created 3 data sets by randomly selecting equal number of positive and negative instances from the pool of the original positive and negative instances.

To find the relationship between the length of the surrounding TATA box and its influence on discriminating TATA from putative TATA boxes, we have collected positive and negative instances of the data for substrings of length 6, 9, 12 and 15.

3.1 Application of Extension 1

As had been illustrated in section 2, the prior probabilities of classes also influence the outcome. The prior probabilities of classes are computed from the training set. To minimize the bias on the computed prior probability from training instances, we perturb the value so as to maximize some objective function, which is some combination of prediction accuracy and the separation between the true positive and false positive, which is denoted by λ PA + (1- λ)(TP – FP) where $\lambda \in [0..1]$.

We have created a training set corresponding to the flanking sub sequences to 174 tata boxes and 459 putative tata boxes. The negative instances are 2.64 times of the positive instances. We use 10 fold cross validation, that is, the data set was randomly divided into 10 equal parts and the algorithm was trained with all but one partition,

Table 2. Performance of applying extension one to naïve Bayes. The average of running 10 fold cross validation for 5 times.

	Offset 6	Offset 9	Offset 12	Offset 15
PA	0.717	0.724	0.735	0.737
TP	0.539	0.595	0.620	0.630
FP	0.216	0.227	0.222	0.222
λ PA + (1- λ)(TP – FP)	0.618	0.636	0.650	0.655

Table 3. Performance of applying naïve Bayes. The average of running 10 fold cross validation running 5 times.

	Offset 6	Offset 9	Offset 12	Offset 15
PA	0.735	0.738	0.753	0.739
TP	0.360	0.413	0.468	0.474
FP	0.122	0.138	0.139	0.160
λ PA + (1- λ)(TP – FP)	0.611	0.622	0.647	0.633

which was used for testing. The training and testing were repeated 10 times so that each partition was tested exactly once. It has been empirically shown [7] that 10 fold cross validation gets the best estimation of error. During training, the prior probability is perturbed to maximize the objective function with λ set to 0.75 and then the model is used to predict the classes in the test set. The results in table show the average of running 10 fold validation 5 times. For comparison we ran 10 fold cross validation of naïve Bayes 5 times and the results are shown in table 3.

The graphs in Figure 1 show the comparison of naïve Bayes with and without extension. The naïve Bayes without the extension performed slightly better that that of with the extension, but there is a significant difference between their true positive rates. When we compare the false positive rate in table 1 and 2, we notice that false positive rate was lower with the large number of negative training instances in naïve Bayes due to bias towards negative instances. On the other hand, the extension improved the true positive. Further, we can observe that the objective function that combines the prediction accuracy and the difference between the true positive and false negative is more or less same for both naïve Bayes with and without extension.

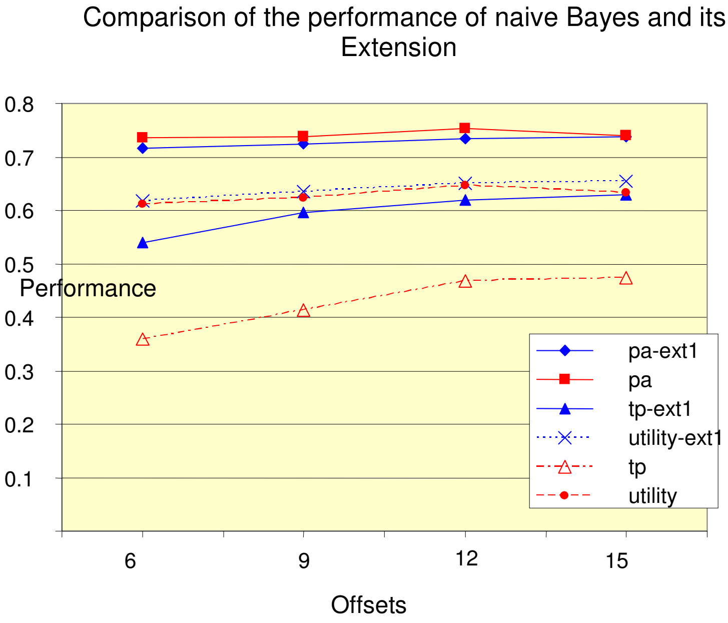


Fig. 1. Comparing the performance with and without the extension

3.2 Application of Extension 2

We test the effectiveness of limited dependency in improving the prediction accuracy as well as maximizing the difference between the true positive and false positive. We have developed the algorithm and have implemented it in Perl to train and test the limited dependency Bayes extension. For testing, we have used 10 fold cross validation.

We have three data sets each has equal number of positive and negative instances. We ran 10 fold cross validation 5 times for each data set and computed the prediction accuracy, true positive false positive by taking the average of the 5 runs. For the combined outcome of prediction accuracy with the difference between true positive and false positive we have used λ=0.75. The results are shown in Tables 4 through 6. A graphical comparison of prediction accuracy with different Markov order is shown in Figure 2 from the data in Table 4.

For comparing the results with that of using artificial neural network, we have used Weka [5]. In our previous work [13] we compared the prediction performance of different algorithms for their effectiveness of discriminating putative TATA boxes from TATA boxes and these algorithms include Naïve Bayes, artificial neural network, random forest, Decision tree C4.5 and Support Vector Machine. We found that the artificial neural network outperformed the rest of the algorithms and the best results occurred at the offset 12. It has the highest prediction accuracy of 78% with the true positive rate of 73% and false positive rate of 17% while the naïve bayes has the prediction accuracy of 65.5% and true positive rate of 63.8%. The limited dependency extension of naïve bayes has outperformed even the best performance of

Table 4. Prediction accuracy variation with offset and partial dependency

Markov order	Offset 6	Offset 9	Offset 12	Offset 15
1	63.91%	65.57%	65.54%	68.72%
2	67.41%	72.70%	74.98%	76.65%
3	72.28%	79.10%	79.56%	79.92%
4	77.07%	75.88%	77.34%	79.20%
5	77.41%	76.19%	77.62%	77.51%

Table 5. The variation of performance metrics with offset and dependency. Here the column *Diff* denote for the (true positive – false positive).

Markov order	Offset 6			Offset 9			Offset 12			Offset 15		
	TP	TN	Diff	TP	TN	Diff	TP	TN	Diff	TP	TN	Diff
1	60.6	67.2	27.8	63.3	67.9	31.1	63.8	67.3	31.1	67.1	70.3	37.4
2	65.2	69.6	34.8	71.0	74.4	45.4	74.6	75.3	50.0	78.2	75.1	53.3
3	70.7	73.9	44.6	79.4	78.8	58.2	84.0	75.1	59.1	87.6	72.2	59.8
4	75.1	79.0	54.1	72.6	79.2	51.8	77.9	76.8	54.7	82.0	76.4	58.4
5	73.1	81.7	54.8	72.3	80.1	52.4	76.0	79.3	55.2	76.4	78.6	55.0

Table 6. The variation of utility with λ=0.75 with Offset and dependency

Markov Order	offset6	offset9	offset12	offset15
1	54.89%	56.97%	56.92%	60.90%
2	59.27%	65.88%	68.73%	70.81%
3	65.35%	73.87%	74.45%	74.90%
4	71.34%	69.85%	71.67%	73.99%
5	71.77%	70.23%	72.03%	71.89%

Artificial netral network; the extension with dependency 2 has the best average performance of 79.9% with the true positive rate of 87.6%.

The graphs in Figure 2 show the prediction accuracy for different Markov order 0 to 4 with order 0 reduces to naïve Bayes without any extension. We can notice a big improvement in prediction accuracy even with a single level of dependency. For this particular problem, order 2 seems to have the best prediction accuracy compared with other dependency levels. The order level 3 and 4 behave very similar. In our previous work, we have compared the prediction performance of several machine learning algorithms including artificial neural networks, random forests, decision trees and naïve Bayes and for the comparison we reproduce the results in Figure 3. Further, this extension with order 2 has outperformed that of many other machine learning algorithms.

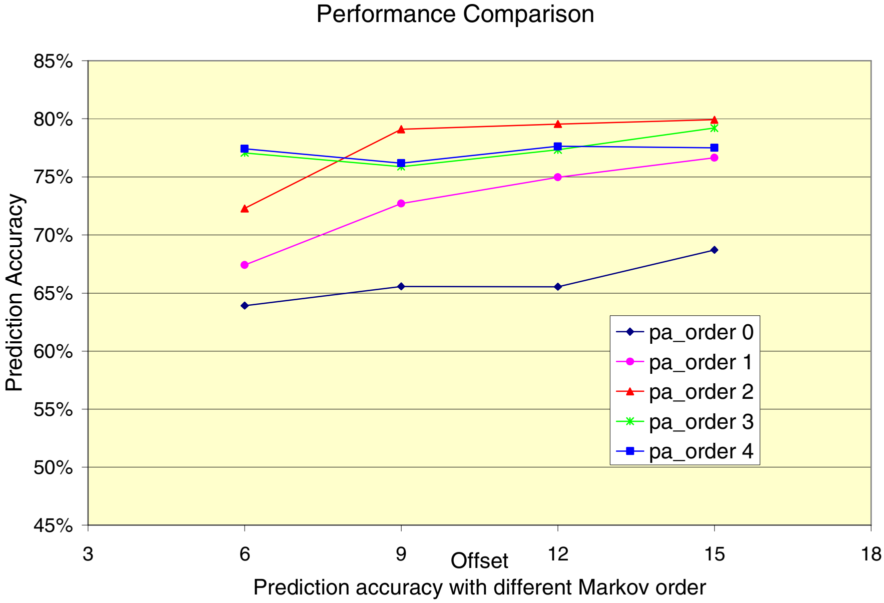


Fig. 2. Comparison of prediction accuracy for different Markov order

The predication accuracy is one of the performance matrices used to compare the performance of machine learning algorithm. The other important metric of performance measurement is the difference between the true positive and the false positive. We have defined an objective function that combine the prediction accuracy with the difference between the true positive and the false positive. We show in Figure 3 the variation of the objective function, combined utility, with the Markov order. The performance on the combined utility with dependency 2 has outperformed that of other extensions similar to what it has done with other performance metrics.

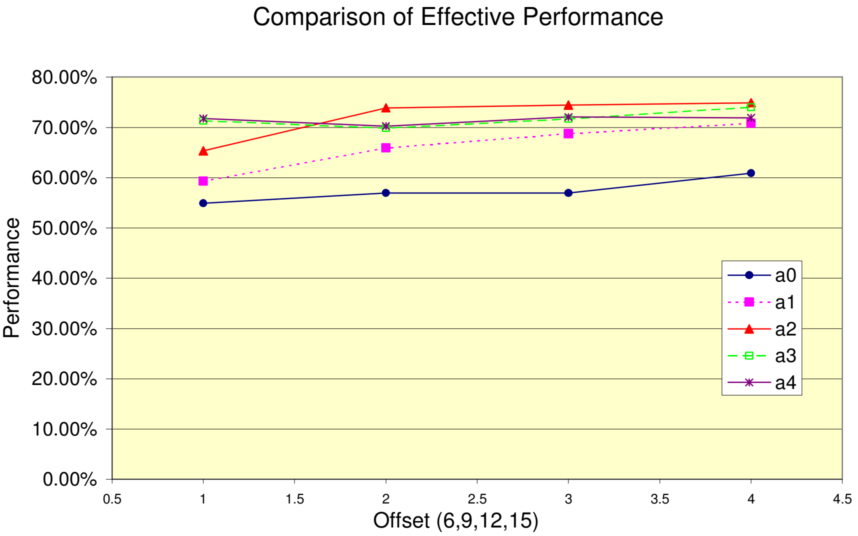


Fig. 3. Objective fn 0.75(PA)+0.25(TP-FN) for different Markov Order

4 Summary and Conclusion

Naïve Bayes, a crude approximation to Bayesian network, has been used very successfully as a classifier to solve many problems spanning from bioinformatics to text mining. Naïve Bayes assumes positional independence, which makes the computation of the joint probability vale easier at the expense of the accuracy or the underlying reality. Further, Naïve Bayes compute the prior probability of the classes from the instances in the training set, which usually does not reflect the correct prior probability of classes and thus bias the decision of the classifier. In this paper we have addressed these two problems and have proposed methods and algorithms to improve the prediction problem of naïve Bayes.

To compare the performance of the proposed extensions and to compare with other algorithms, we use prediction accuracy and as well as a combination of prediction accuracy with the difference between true positive and false positive. We have proposed a method in section 2 to improve the computation of prior probability of classes and to avoid possible bias on classification due to the instances of training set. The method is based on perturbing prior probability around pre computed value so as to maximize the objective function, which was the combination of 0.75 of prediction accuracy and 0.25 of the differences between true positive and false positive. The 0.75 and 0.25 combinations are arbitrary and we have used these values for illustration purpose. Once the best value of the prior probability was found, the value was used to classify the test along with other trained conditional probability. To find the effectiveness of this extension, we have applied it to solve a problem that we have worked before, discriminating TATA box from putative TATA box. The results are shown in Tables 2 and 3 and in Figure 1. In this experiment we have used 10 fold cross validation on a data set with 174 positive instances and 459 negative instances.

If we look at the comparison of the prediction accuracy of naïve Bayes with and without the extension in Figure 1, they looked very much similar and in fact the naïve Bayes seems to have a better performance, which can be explained due to increased value of true negative sacrificing true positive. Also notice that the difference between true positive and false positive remains more or less same with naïve Bayes with and without extension.

To improve the prediction accuracy we introduce limited dependency in the neighborhood. The order of dependency varies from 0 to k where k is the length of the neighbor hood. As the number of dependency increases the prediction accuracy may start increasing and then start dropping due to data over fitting. Empirically we have shown that the prediction accuracy with Markov order 2 achieves the best performance. The best prediction accuracy of a naïve Bayes with 10 fold cross validation was 69% while the extension gave the prediction accuracy of 79% which is better than the best solution by an artificial neural network. The extension achieves the improved prediction accuracy without the increase in computational cost and it is relatively easy to implement compared to many other sophisticated machine learning algorithms such as artificial neural network. When applying this extension to solve a new problem, find the appropriate level of dependency that has the best prediction accuracy and use the level of dependency for testing purposes.

In summary we have introduced two extensions to naïve Bayes to improve its predictability without increasing the computational cost. Empirically we have shown that the second extension has outperformed other best known machine learning algorithms. To make very firm conclusion of the effectiveness of these extensions, we need to apply this methods to many other problems and empirically show the performance improvements.

References

1. Cao J, Panetta R, Yue S, Steyaert A, Young-Bellido M, Ahmad S: **A naive Bayes model to predict coupling between seven transmembrane domain receptors and G-proteins**. *Bioinformatics* 2003, **19**(2):234-240.
2. Ferrari LD, Aitken S: **Mining housekeeping genes with a Naive Bayes classifier**. *BMC Genomics* 2006, **7**(277).
3. Sandberg R, Winberg G, Bränden C-I, Kaske A, Ernberg I, Cöster J: **Capturing Whole-Genome Characteristics in Short Sequences Using a Naïve Bayesian Classifier**. *Genome Research* 2001, **11**(8):1404-1409.
4. Wu J, Mellor JC, DeLisi C: **Deciphering protein network organization using phylogenetic profile groups**. *Genome Inform* 2005, **16**(1):142-149.
5. **Weka**. In.; 2006: Data Mining Software in Java, http://www.cs.waikato.ac.nz/ml/weka/.
6. Jaynes ET: **Prior Probabilities**. *IEEE Trans on Systems Science and Cybernetics* 1968, **Sec-4**(3):227-241.
7. Witten IH, Frank E: **Data mining: practical machine learning tools and techniques**, second edn. San Francisco, Calif.: Morgan Kaufmann; 2005.
8. Tompa M, Li N, Bailey TL, Church GM, De Moor B, Eskin E, Favorov AV, Frith MC, Fu Y, Kent WJ *et al*: **Assessing computational tools for the discovery of transcription factor binding sites**. *Nat Biotechnol* 2005, **23**(1):137-144.

9. Butler JEF, Kadonaga JT: **The RNA polymerase II core promoter: a key component in the regulation of gene expression**. *Genes Development* 2002, **16**:2583-2592.
10. Loganantharaj R, Karim ME, Lakhotia A: **Recognizing TATA promoters based on discriminating frequency analysis of neighborhood tuples**. In: *Biot-04: First Biotechnology and Bioinformatics Symposium: A Community and Academic Forum: 2004, September; Colorado Springs, Colorado*; 2004, September.
11. Loganantharaj R: **Discriminating TATA-box from putative TATA box in plant genome**. *International Journal of Bioinformatics Research and Applications* 2006, **2**(1):36-51.
12. Shahmuradov IA, Gammerman AJ, Hancock JM, Bramley PM, Solovyev VV: **PlantProm: a database of plant promoter sequences**. *Nucleic Acids Res* 2003, **31**(1):114-117.
13. Logananatharaj R: **Comparing the Performance of Several Popular Machine Learning Algorithms**. In: *Biotechnology and Bioinformatics Symposium: October, 20-21 2006; Provo, Utah*; 2006.

The Solution Space of Sorting by Reversals

Marília D.V. Braga[1], Marie-France Sagot[1], Celine Scornavacca[2], and Eric Tannier[1]

[1] INRIA Rhône-Alpes, Laboratoire de Biométrie et Biologie Évolutive (UMR 5558),
CNRS, Univ. Lyon 1
43 bd 11 Nov, 69622, Villeurbanne Cedex, France
marilia@biomserv.univ-lyon1.fr,
{Marie-France.Sagot,Eric.Tannier}@inrialpes.fr

[2] Laboratoire d'Informatique, de Robotique et de Microélectronique
de Montpellier, 34392 Montpellier Cedex 5 - France
Celine.Scornavacca@lirmm.fr

Abstract. In comparative genomics, algorithms that sort permutations by reversals are often used to propose evolutionary scenarios of large scale genomic mutations between species. One of the main problems of such methods is that they give one solution while the number of optimal solutions is huge, with no criteria to discriminate among them. Bergeron *et al.* [4] started to give some structure to the set of optimal solutions, in order to be able to deliver more presentable results than only one solution or a complete list of all solutions. The structure is a way to group solutions into equivalence classes, and to identify in each class one particular representative. However, no algorithm exists so far to compute this set of representatives except through the enumeration of all solutions, which takes too much time even for small permutations. Bergeron *et al.* [4] state as an open problem the design of such an algorithm. We propose in this paper an answer to this problem, that is, an algorithm which gives one representative for each class of solutions and counts the number of solutions in each class, with a better theoretical and practical complexity than the complete enumeration method. We give several biological examples where the result is more relevant than a unique optimal solution or the list of all solutions[1].

1 Introduction

The combinatorics of genome rearrangements is a very prolific domain of computational biology. It consists in, given a set of actual genomes, inferring the large-scale evolutionary mutations that explain the differences in the organisation of those genomes. For a general survey of the algorithmic aspects of genome rearrangements, see [13].

One of the most used mathematical models for representing and manipulating such genome rearrangements is given by signed permutations, where the elements

[1] An implementation of the algorithm is available online, as part of the BaobabLuna package, at www.geocities.com/mdvbraga/baobabLuna.html

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 293–304, 2007.

are unique homologous markers, and reversals as the main events that may alter the order of the markers along the genomes. The combinatorial problem consists then in giving a shortest sequence of reversals that transforms one permutation into another. The problem of sorting signed permutations by reversals has been the subject of a huge literature (among others, [2,5,17,12,8,11]), but all algorithms propose one optimal solution, whereas the solutions can be very numerous. This kind of delivery may be useless for biological purposes, and the algorithms are therefore mainly useful to compute a distance between genomes.

One study by Siepel [14] resulted in a method to enumerate all solutions. This is however almost as useless as providing only one solution, because often the solutions are so many that the whole set can not be presented (when it can be computed). A few studies tried to decrease the size of the set of optimal solutions by introducing some biological constraints, such as favouring small inversions [1], or inversions that do not cut some clusters of co-localised genes [8,3]. The number of solutions is decreased, but the whole set of solutions is never handled.

Bergeron *et al.* [4] then provided a way to group the solutions into equivalence classes. However, no algorithmic study was performed, and in particular the problem of giving one element in each class without enumerating all the solutions was mentioned open. In this paper, we introduce a solution to this problem. Our solution gives one representative element per class of solutions, and counts the number of solutions in each class. The complete enumeration of the solutions is not needed, and the theoretical complexity, as well as the practical execution time are lower than in any other current method for the enumeration of the solutions.

The paper is organised as follows. We present the usual model for dealing with gene order and orientation in Section 2. In Section 3, we describe the algorithm. Section 4 is dedicated to practical experiments on simulated and biological data, and on an analysis of the performances of our implementation.

2 Sorting by Reversals and Its Solution Space

Signed permutations. Genome rearrangements such as reversals may change the order of some segments in a genome, and also the DNA strand the segment is on. We identify homologous genomic markers with the integers $1, \ldots, n$, with a plus or minus sign to indicate the strand they lie on. The order and orientation of genomic markers of one species in relation to another is represented by a *signed permutation* of size n, that is, by a permutation of the set $\{1, \ldots, n\}$, where each number is, in addition, given a sign '+' or '-' (the sign '+' is usually omitted). The *identity permutation* $(1, \ldots, n)$ is denoted by Id.

A subset of numbers $\rho \subseteq \{1, \ldots, n\}$ is said to be an *interval* if there exist $i, j \in \{1, \ldots, n\}$, $1 \leq i \leq j \leq n$, such that $\rho = \{|\pi_i|, \ldots, |\pi_j|\}$. Two intervals are said to *overlap* if they intersect but none is contained in the other.

Sorting by reversals. Given a permutation π and an interval ρ, we can apply a *reversal* on π, that is, the operation which reverses the order and flips the signs of the elements of ρ: if $\rho = \{|\pi_i|, \ldots, |\pi_j|\}$,

$$\pi \cdot \rho = (\pi_1, \ldots, \pi_{i-1}, -\pi_j, \ldots, -\pi_i, \pi_{j+1}, \ldots, \pi_{n+1}).$$

Due to this, an interval ρ can also be used to denote a reversal. We may always represent an interval or reversal ρ as the sorted set of its values.

If $\rho_1, \ldots, \rho_k$ is a sequence of intervals or reversals, we say that it *sorts* a permutation π if $\pi \cdot \rho_1 \cdots \rho_k = Id$. The length of a shortest sequence of reversals sorting a permutation π is called the *reversal distance* of π, and is denoted by $d(\pi)$. A shortest sequence of reversals sorting π is called an *optimal* sorting sequence. For example, if the permutation π is $(4, -3, -1, 2)$, one optimal sorting sequence is $\{1\}, \{1, 2\}, \{4\}, \{1, 2, 3, 4\}$.

Computing the reversal distance and finding an optimal sorting sequence has been the topic of a huge literature. The first polynomial algorithm appeared in [12], while the fastest algorithm to compute the distance was given in [2], and the one to find an optimal sequence can be retrieved from a compilation of [5,11,17].

However, all these studies give one sequence among possibly many. For example, for the permutation $(-12, 11, -10, 6, 13, -5, 2, 7, 8, -9, 3, 4, 1)$, the number of solutions is 8278540, and it can be useless when attempting a biological interpretation to know only one among them.

The set of all solutions may be retrieved thanks to an algorithm by Siepel [14], that, given a permutation, computes all the reversals that are the first step of an optimal sequence, but in the aforementioned example, listing the 8278540 sequences is almost as useless as giving only one of them.

Traces. More interesting for our study is the representation of the set of solutions that is given in [4]. Recall a reversal is written as a subset of $\{1, \ldots, n\}$, where the elements are ordered increasingly, so that they can be compared by a lexicographic order. Identifying a sequence of reversals with a word on the alphabet $\mathcal{A}$ of reversals, the authors of [4] define an equivalence relation on these words: if ρ and θ are reversals (intervals) and do not overlap, then the words $\rho\theta$ and $\theta\rho$ are equivalent. We say that ρ and θ *commute*. Under this relation, any two words containing $\rho\theta$ as a subword are equivalent to the same word, replacing the subword $\rho\theta$ by $\theta\rho$.

For example, if the permutation π is $(4, -3, -1, 2)$, consider the solution given by the sequence of reversals $\{1\}\{1, 2\}\{4\}\{1, 2, 3, 4\}$. Here, $\{4\}$ and $\{1, 2\}$ commute, so $\{1\}\{1, 2\}\{4\}\{1, 2, 3, 4\}$ is equivalent to $\{1\}\{4\}\{1, 2\}\{1, 2, 3, 4\}$, and as every pair of reversals also commute, every permutation of these four reversals is a solution.

Now if the solution $\{1, 3, 4\}\{2, 4\}\{2, 3\}\{3\}$ is considered, then it is equivalent to $\{1, 3, 4\}\{2, 4\}\{3\}\{2, 3\}$, $\{1, 3, 4\}\{3\}\{2, 4\}\{2, 3\}$ and $\{3\}\{1, 3, 4\}\{2, 4\}\{2, 3\}$ by commutation of $\{3\}$ with all the other reversals, which do not commute among themselves.

An equivalence class of optimal sequences of reversals over this equivalence relation is called a *trace*. The concept of traces is well studied in combinatorics, see for example [7]. It is particularly relevant in our study because of the following result proven in [4].

Proposition 1. [4] *Let π be a signed permutation. The set of all optimal sequences of reversals sorting π is a union of traces.*

As a consequence, if the set of solutions is too big to be enumerated, the set of traces may be a more relevant result for the problem of sorting by reversals. It remains to find a good way to represent the traces in a compact manner.

Normal form of a trace. A trace T is thus a set of equivalent words over an alphabet $\mathcal{A}$. An element s of T is said to be in *normal form* if it can be decomposed into subwords $s = u_1|\ldots|u_m$ such that:

- every pair of elements of a subword u_i commute;
- for every element ρ of a subword u_i $(i > 1)$, there is at least one element θ of the subword u_{i-1} such that ρ and θ do not commute;
- every subword u_i is a nonempty increasing word under the lexicographic order induced by $\mathcal{A}$

A theorem by Cartier and Foata (cited in [4]) states that, for any trace, there is a unique word that is in the normal form. We may therefore represent a trace by its element in the normal form.

For example, the permutation $(4, -3, -1, 2)$ has two traces of optimal sequences, one is $\{1\}\{1,2\}\{1,2,3,4\}\{4\}$, and the other is $\{1,3,4\}\{3\}|\{2,4\}|\{2,3\}$. In this example, giving the two normal forms of the traces allows to describe the whole set of 28 solutions in a compact way.

The algorithmics of traces. Bergeron *et al.* [4] provide no algorithmic insight for this way of representing the solutions of sorting by reversals. They state as an open problem the complexity of giving one element in each trace. The best algorithm so far to enumerate the traces is therefore to do a complete enumeration of all the solutions, and from each solution, to compute the associated trace and add it to the list of found traces if it is not already in.

We give in this paper an algorithm that enumerates the normal form of all the traces of solutions given a signed permutation, and counts the number of solutions in each trace, without enumerating all the solutions.

3 The Algorithm and Its Complexity

It will be useful to describe first the only available algorithm that is up to now able to enumerate all the traces of the solution space of sorting by reversals, and to examine its theoretical complexity. We then present our algorithm, and make a comparison between the two.

3.1 The Enumeration of the Solutions

A sequence of reversals $s = \rho_1\rho_2\ldots\rho_i$ is called an *optimal i-sequence* if $d(\pi \cdot \rho_1 \cdots \rho_i) = d(\pi) - i$. Note that if $i = d(\pi)$, then s is an optimal sorting sequence.

The set of all optimal 1-sequences of a permutation can be computed with the help of an algorithm by Siepel [14]. It has time complexity $O(n^3)$, and the number of possible optimal 1-sequences is bounded by $\frac{n(n+1)}{2} \leq n^2$.

The set of all optimal i-sequences can then be computed from the set of $(i-1)$-sequences by iterating the same algorithm for finding all 1-sequences. The set of i-sequences has therefore size at most $O(n^{2i})$, and the algorithm has time complexity at most $O(n^3 * \sum_{k=1}^{i} n^{2k})$. In this way, we can enumerate the set of all optimal sorting sequences in time $O(n^{2n+3})$.

There remains to construct the normal form of the trace for each sorting sequence, and then to group the sorting sequences by trace.

For any optimal i-sequence s of reversals, and under the equivalence relation deduced from the commutation of reversals, is defined the trace that contains s, that we call an *i-trace*.

Given an optimal sorting sequence $s = \rho_1\rho_2\ldots\rho_d$ for a permutation π with reversal distance d, the normal form of the trace T that contains s is constructed by iterating an integer i from 1 to d and, at each step i, adding the element ρ_i, represented as the sorted set of its values, to the normal form of the $(i-1)$-trace containing $\rho_1\ldots\rho_{i-1}$ (the initial 0-trace is an empty trace). This procedure is described by Algorithm 1.

Algorithm 1. Adding an element to a normal form of a trace

```
Require: An $(i-1)$-trace $u_1|u_2|\ldots|u_k$ and the next element $\rho_i$
Ensure: The normal form of the $i$-trace containing the element $u_1u_2\ldots u_k\rho_i$

  Let $j$ be the maximum index such that $u_j$ contains an element that does not commute
  with $\rho_i$, or 0 if such a $u_j$ does not exist
  if $j = k$ then
    Add a new subword $u_{k+1} \leftarrow \rho_i$
  else
    Add $\rho_i$ to the subword $u_{j+1}$, according to the lexicographic order
  end if
```

As the reversal distance and the interval size are bounded by n, the procedure has complexity $O(n^2 \log n)$, considering that each reversal or interval has to be sorted and comparing reversals may be done in $O(n)$.

The constructed solution is compared to a list of already constructed normal forms of traces, so that one trace is not written several times. This may take $O(n \log N)$ operations, where N is the number of represented traces. As N is bounded by the number of solutions, we have $n \log N \leq n \log(n^{2n}) = 2n^2 \log n$.

Eventually, the total time complexity for enumerating all the normal forms of the traces is bounded by $O(n^{2n+3}) + O(n^{2n}(n^2 \log n + 2n^2 \log n)) = O(n^{2n+3})$.

This upper bound on the theoretical complexity does not give hope that this method can be applied to big permutations. We shall actually see in practice that it is intractable for permutations π above around $d(\pi) = 10$.

This method is implemented, for example, in the GRAPPA software[2], and it is the only one that, among all available applications about sorting by reversals, is able to give more than one unique solution.

3.2 The Enumeration of the Traces

A k-trace T' is a *prefix* of an i-trace T $(k \leq i)$ if T' contains a k-sequence which is a prefix of an i-sequence of T. It is equivalent [7] to saying that each k-sequence of T' is a prefix of an i-sequence of T.

The idea of the algorithm to enumerate the traces is almost naturally contained in this notion. It is easy to remark that every prefix of size k of an optimal i-sequence is in a k-trace of optimal k-sequences. So instead of enumerating all the i-sequences and then computing and comparing the traces, it is therefore more valuable to enumerate and compare directly all the i-traces.

We have seen in Algorithm 1 a way to construct the normal form of an $(i+1)$-trace from the one of an i-trace. We may use this method to construct all i-traces simultaneously in an incremental way, without computing all the solutions. With no additive cost, we also compute the number of sequences in each i-trace.

The method is detailed in Algorithm 2.

Theorem 1. *At the end of Algorithm 2, $\mathcal{T}$ contains, for every trace T of solutions for sorting π, one element of T (the normal form) and the number of solutions in T.*

Proof
The proof is by induction. We prove that at the end of the step i of the main loop of Algorithm 2, the set $\mathcal{T}$ contains all the normal forms and the size of the i-traces of optimal sequences for π.

For $i = 1$, each 1-trace is generated by the algorithm of Siepel [14] and the size of a 1-trace is 1.

For an arbitrary $2 \leq i \leq d(\pi)$, by hypothesis, $\mathcal{T}$ contains all the normal forms and the size of the optimal $(i-1)$-traces. Every i-trace has a prefix in this set, since a prefix of size $i-1$ of an optimal i-sequence is an optimal $(i-1)$-sequence. So every i-trace is found from an $(i-1)$-trace by the algorithm of Siepel [14].

Now it remains to prove that the cardinality of an i-trace T is the sum of the cardinalities of its $(i-1)$-prefixes, so that the right size of all traces are computed. Let $\rho_1, \ldots, \rho_k$ be the reversals that are in the last position of at least one element in T. Let x_j be the number of elements of T which have ρ_j as their last position. Then the number of elements of T is $\sum_j x_j$. Now, for all j, as ρ_j is the last reversal of an optimal i-sequence $x_1 \ldots x_{i-1}\rho_j$ of T, $x_1 \ldots x_{i-1}$ is an optimal $(i-1)$-sequence of reversals, so it belongs to an $(i-1)$-trace T' of size x_j. So by the induction hypothesis, the size of the trace T is the sum of the sizes

[2] http://www.cs.unm.edu/~moret/GRAPPA/. We re-implemented the algorithm in Java in order to include it in the package BaobabLuna, that implements all the methods that we describe here, in order to compare the running times on the same basis.

Algorithm 2. Enumerating all the traces of a signed permutation

```
Require: A signed permutation $\pi$
Ensure: The normal form and size $(norm(T), size(T))$ of each trace $T$ of optimal
  sequences of reversals for sorting $\pi$

  $d \leftarrow$ reversal distance of $\pi$
  $S \leftarrow \{\rho \mid \rho$ is an optimal 1-sequence for $\pi\}$/* Algorithm of Siepel [14] */
  $\mathcal{T} \leftarrow \emptyset$
  for each reversal $\rho \in S$ do
    $norm(T) \leftarrow \rho$ /* $T$ is a 1-trace */
    $size(T) \leftarrow 1$
    Insert $\{(norm(T), size(T))\}$ in $\mathcal{T}$
  end for
  for each integer $i$ from 2 to $d$ do
    $\mathcal{T}_{next} \leftarrow \emptyset$ /* contains the normal forms of all the $i$-traces */
    for each $(norm(T), size(T))$ in $\mathcal{T}$ /* $T$ is a $(i-1)$-trace */ do
      Let $\pi_T$ be the resulting permutation after applying the $(i-1)$-sequence $norm(T)$
      to $\pi$
      $S \leftarrow \{\rho \mid \rho$ is an optimal 1-sequence for $\pi_T\}$/* Algorithm of Siepel [14] */
      for each reversal $\rho \in S$ do
        $norm(T_{+\rho}) \leftarrow norm(T) + \rho$ /* Algorithm 1 */
        $size(T_{+\rho}) \leftarrow size(T)$
        if there is $(norm(T_+), size(T_+)) \in \mathcal{T}_{next}$ such that $norm(T_+) = norm(T_{+\rho})$
        then
          $size(T_+) \leftarrow size(T_{+\rho}) + size(T_+)$
        else
          Insert $(norm(T_{+\rho}), size(t_{+\rho}))$ in $\mathcal{T}_{next}$
        end if
      end for
    end for
    $\mathcal{T} \leftarrow \mathcal{T}_{next}$
  end for
  return $\mathcal{T}$ /* $\mathcal{T}$ is the final set of $d$-traces */
```

of all $(i-1)$-prefixes of T, and the algorithm provides this size, since it generates all prefixes. □

3.3 Theoretical Complexity

The complexity of the algorithm depends on the number $\sum_{i=1}^{d(\pi)} n(i)$, where $n(i)$ is the number of i-traces of optimal i-sequences. As every i-trace is a prefix of a d-trace, where $d = d(\pi)$, this number is bounded by the number of d-traces times the number of prefixes of each trace.

To give an estimation of the number of prefixes of a trace, we need to adopt a representation of the traces as partially ordered sets (posets). It is possible to represent a trace T that contains an optimal sequence $\rho_1 \dots \rho_n$ by a partial ordering of the set $\mathcal{P}_T = \{(\rho_i, k_i)\}_i$, where ρ_i is an element of $\mathcal{A}$ appearing in

$\rho_1 \dots \rho_n$ and k_i is the number of occurrences of ρ_i in the subword $\rho_1 \dots \rho_i$. The relation $<_T$ is defined as the transitive closure of the relation $\lhd$ itself defined by $(\rho_i, k_i) \lhd (\rho_j, k_j)$ if and only if $i < j$ and ρ_i and ρ_j do not commute.

In other words, $(\rho_i, k_i) <_T (\rho_j, k_j)$ if and only if the k_i^{th} ρ_i is always before the k_j^{th} ρ_j in the elements of T (see [7]).

For example, $T = \{1,3,4\}\{3\}|\{2,4\}|\{2,3\}$ is a trace of optimal sequences for the permutation $(4,-3,-1,2)$. The elements of $\mathcal{P}_T$ are $(\{1,3,4\},1)$, $(\{3\},1)$ $(\{2,4\},1)$ and $(\{2,3\},1)$, and the relations are $(\{1,3,4\},1) <_T (\{2,4\},1)$, $(\{2,4\},1) <_T (\{2,3\},1)$ and $(\{1,3,4\},1) <_T (\{2,3\},1)$. The poset is represented in Figure 1.

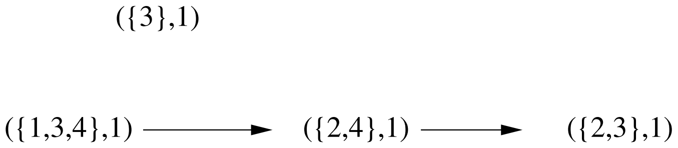


Fig. 1. The poset constructed from the normal form $T = \{1,3,4\}\{3\}|\{2,4\}|\{2,3\}$. All the linear extensions of this poset are the optimal sequences of reversals belonging to the trace represented by T.

The set $\mathcal{P}_T$ with the relation $<_T$ is a partially ordered set (poset). A *linear extension* of a poset is a total order $<_{tot}$ which verifies $\rho <_T \theta \Rightarrow \rho <_{tot} \theta$. The set of all linear extensions of $\mathcal{P}_T, <_T$ is exactly the set of elements of the trace T (see [7]). We may therefore identify the trace T and the poset $\mathcal{P}_T, <_T$, and simply speak about the poset T.

The *height* of a trace (or poset) is the cardinality of the maximum set of elements of $\mathcal{P}_T$ that is totally ordered by the relation $<_T$. It is also the number of subwords u_i in the normal form of a trace.

The *width* of a trace (or poset) is a maximum cardinality set of elements of $\mathcal{P}_T$ that are not comparable by the relation $<_T$. It is at least (but in general not equal to) the maximum size of a subword u_i in the normal form of a trace. The width of a poset can be computed in polynomial time thanks to a reduction of Fulkerson [10] to a bipartite matching problem.

The representation of a trace as a poset allows to use the parameters of the poset in the computations of the complexity of the algorithms, and it is also a nice way to present the solution of sorting by reversals. Indeed, a poset corresponds to a set of reversals that may have occurred during evolution and that could therefore help explain the difference between the organisation of two genomes. It indicates what we know and what we do not know about the order in which these potential reversals occurred. Instead of giving a list of sequences, or a unique sequence representing an equivalence class, the poset therefore gives *one* possible solution, with uncertainties as concerns the exact shape of the solution.

An *ideal* of a poset $\mathcal{P}_T, <_T$ is a subset U of $\mathcal{P}_T$ such that if $\rho \in U$ and $\theta <_T \rho$, then $\theta \in U$.

It is very easy to see that ideals of posets and prefixes of traces correspond to the same notions, and that in particular, the number of prefixes of a trace T is exactly the number of ideals of the poset $\mathcal{P}_T, <_T$.

The advantage of this notation is that the number of ideals of a poset can be estimated. It is bounded by n^k, where n is the size of $\mathcal{P}_T$ and k is the width of the poset [15].

The number of i-traces that we generate is therefore bounded by $Nn^{k_{max}}$, where N is the number of d-traces and k_{max} is the maximum width of a d-trace.

Given this estimation, we may give a bound for the complexity of our algorithm. Indeed, for every i-trace, $1 \leq i \leq d-1$, we apply an $O(n^3)$ algorithm to find all the 1-sequences. For all these 1-sequences (there are at most n^2 of them), we then apply Algorithm 1 to construct the normal form of the following $(i+1)$-trace, and compare the constructed normal form to the current list of normal forms of $(i+1)$-traces.

This gives a final complexity of $O(Nn^{k_{max}}(n^3 + n^2(n^2 + n \log Nn^{k_{max}}))) = O(Nn^{k_{max}+4})$.

Observe that computing the number of linear extensions of a poset is $\#P$-complete [6], and the best known algorithms run in $O(n^k)$, where n is the size of the poset and k is its width [16]. Our algorithm counts the number of elements in each d-trace, that is the number of linear extensions of the associated posets. Our time complexity thus nearly reaches the best known complexity for the counting part.

If in general the width of a poset may be as large as its number of elements, we have made some experiments on simulated permutations (see Figure 2) which show that in practice, this parameter is often lower, which explains the speed-up of our algorithm compared to a total enumeration procedure.

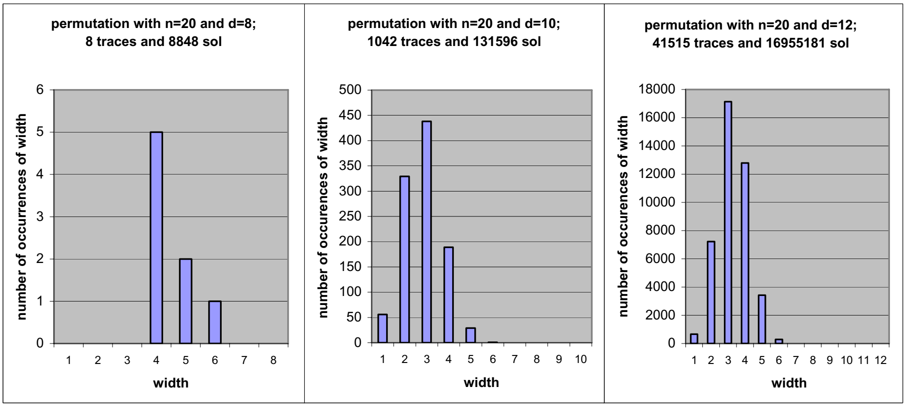


Fig. 2. Distribution of the width of posets on random permutations

4 Experimental Results and Applications

We implemented our algorithm and tested it on randomly simulated permutations, as well as on biological data[3]. Some results are recorded in Figure 3. These numbers may be useful to give an idea of the quantities that we are dealing with, numbers of solutions and number of traces.

Even if we are quickly limited in the size of permutations that are possible to treat, there is a solid gain in relation to the existing methods. Observe that the main limit concerns the amount of memory that needs to be used, more than the time.

5 Conclusions, Limitations and Perspectives

We devised an algorithm that gives a representation of all the solutions of sorting signed permutations by reversals, without enumerating each solution. It is the first algorithm that achieves this, to our knowledge, and it performs better than the complete enumeration on all the data on which we tested it.

PERMUTATION	$\mathbf{N_s}$	$\mathbf{N_t}$	**Enum. sol.**	**Enum. + traces**	**BaobabLUNA**
rat/human chr X n =16 d=10	2419750	418	$\simeq$ 10 min	$\simeq$ 13 min	$\simeq$ 10 s
mouse/human chr X n =16 d=10	3362310	218	$\simeq$ 12 min	$\simeq$ 18 min	$\simeq$ 10 s
random n =17 d=11	57019369	18255	$\simeq$ 4 h	$\simeq$ 4.5 h	$\simeq$ 5 min
random n =18 d=12	327905046	34317	$\simeq$ 24 h	$\simeq$ 43 h	$\simeq$ 18 min
Chr X/Chr Y human n =30 d=12	207600628	115512	$>$ 24 h	$>$ 48 h	$\simeq$ 28 min

Fig. 3. Computation results. Columns from left to right contain: 1- the origin of the permutation, its number of elements and reversal distance; 2- the number of solutions of sorting this permutation by reversals; 3- the number of traces; 4- the execution time of an algorithm that enumerates all the solutions; 5- the execution time of the same program, adding the computation of all the traces from the solutions; 6- the execution time of the enumeration of the traces, according to Algorithm 2.

[3] The first two permutations, that model the organisation of the X chromosomes of human, mouse and rat, are taken from [3]; the permutation `Chr X/Chr Y human`, modelling the comparison of the human X and Y chromosomes, is a simplified version of a set of markers coming from an ongoing study at the LBBE laboratory, University of Lyon 1, France.

The implementation of the algorithm is online, integrated to a package for the manipulation of signed permutations.

Although this program is faster than the previously published methods, it is still limited (mainly because of memory) to small permutations, with $d(\pi)$ inferior to 20, on a personal computer that has 1Gb random access memory[4]. It is sufficient for some biological applications, as we show it here on the data from X chromosomes of mammalian species, or on the comparison of the human X and Y chromosomes (see Figure 3). For many datasets however, it is still insufficient because of the size of the output.

Indeed, if the solution space is dramatically reduced when dealing with traces of solutions, it is often still too big to be handled by biologists on large permutations. The algorithmic limit coincides therefore with the limit of the utility of the solution. Probably another structure remains to be invented in order to solve a similar problem for large permutations.

References

1. Ajana Y., Lefebvre J.F., Tillier E., El-Mabrouk N., "Exploring the set of all minimal sequences of reversals - An application to test the replication-directed reversal hypothesis". Second International Workshop, Algorithms in Bioinformatics (WABI'02), LNCS 2452, R. Guigo and D. Gusfield eds., pp. 300-315, September 2002.
2. Bader, D.A., Moret, B.M.E., and Yan, M., "A linear-time algorithm for computing inversion distances between signed permutations with an experimental study", *J. Comput. Biol.* 8, 5 (2001), 483-491.
3. Berard S., Bergeron A., Chauve C. and Paul C. "Perfect sorting by reversals is not always difficult", to appear in *IEEE transactions on cioinformatics and computational biology*, 2006.
4. Bergeron A., Chauve C., Hartmann T., St-Onge K., "On the properties of sequences of reversals that sort a signed permutation". JOBIM 2002, 99-108.
5. Bergeron A., Mixtacki J. and Stoye J., "The inversion distance problem", in *Mathematics of evolution and phylogeny* (O. Gascuel Ed.) Oxford University Press, 2005.
6. Brightwell G. and Winkler P., "Counting linear extensions is #P-complete", *STOC '91: Proceedings of the twenty-third annual ACM symposium on Theory of computing*, 1991, ACM Press.
7. Diekert V. Rozenberg G. (eds) *The book of traces*, World Scientific, 1995.
8. Diekmann Y., Sagot M.F. and Tannier E., "Evolution under reversals: parsimony and preservation of common intervals", to appear in *IEEE/ACM transactions in computational biology and bioinformatics*, 2006 (A preliminary version appeared in COCOON 2005, *Lecture Notes in Computer Science* 3595, 42-51, 2005).
9. Dilworth R.P., "A Decomposition Theorem for Partially Ordered Sets", Annuals of Mathematics 51 (1950) pp. 161-166.
10. Fulkerson D.R., "Note on Dilworth's decomposition theorem for partially ordered sets", Proc. Amer. Math. Soc. 7 (1956), 701–702

[4] This extensive use of memory is due to the fact that, in order to create the $i-$traces, we have to store all the $(i-1)-$traces.

11. Han Y, “Improving the Efficiency of Sorting by Reversals”, Proceedings of The 2006 International Conference on Bioinformatics and Computational Biology, CSREA Press, Las Vegas, Nevada, USA, 2006.
12. Hannenhalli S. and Pevzner P. , “Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)”, Journal of the ACM, 46:1– 27, 1999.
13. Li Z., Wang L. and Zhang K., “Algorithmic approaches for genome rearrangement: a review”, *IEEE transactions on systems, man and cybernetics*, 36:636–648, 2006.
14. Siepel A. “An algorithm to enumerate sorting reversals for signed permutations”. J Comput Biol 10:575-597, 2003.
15. Steiner G., “An algorithm to generate the ideals of a partial order” Operations Research Letters, 5(6):317 – 320, 1986.
16. Steiner G., “Polynomial algorithms to count linear extensions in certain posets”. Congressus Numerantium, 75, 71-90, 1990
17. Tannier E., Bergeron A. and Sagot M.-F., “Advances on Sorting by Reversals”, to appear in *Discrete Applied Mathematics*, 2006 (A preliminary version appeared in CPM 2004, *Lecture Notes in Computer Science* 3595, 42-51).

A Fast and Exact Algorithm for the Perfect Reversal Median Problem

Matthias Bernt, Daniel Merkle, and Martin Middendorf⋆

Department of Computer Science, University of Leipzig, Germany
{bernt,merkle,middendorf}@informatik.uni-leipzig.de

Abstract. We study the problem of finding for the gene orders of three taxa a potential ancestral gene order such that the corresponding rearrangement scenario has a minimal number of reversals where each of the reversals has to preserve the common intervals of the given input gene orders. Common intervals identify sets of genes that occur consecutively in all input gene orders. The problem of finding such an ancestral gene order is called the perfect reversal median problem (pRMP). A tree based data structure for the representation of the common intervals of all input gene orders is used for the design and realization of a fast and exact algorithm — called TCIP — for solving the pRMP. It is known that for two given gene orders the minimum number of reversals to transfer one gene order into the other can be computed in polynomial time, whereas the corresponding problem with the restriction that common intervals should not be destroyed by the reversals is already NP-hard. Nevertheless, we show empirically on biological and artificial data that TCIP for the pRMP is usually even faster than the fastest exact algorithm (Caprara's median solver) for the reversal median problem (RMP), i.e., the corresponding problem in which the common intervals are not considered.

1 Introduction

The phylogenetic relationship between species is often analyzed by means of rearrangement scenarios for the gene orders of the species. A rearrangement scenario describes how the gene orders can be transferred into each other by a given set of possible rearrangement operations. A very commonly used rearrangement operation is the reversal of a part of the genome. A median for three given signed input permutations is a gene order, such that the sum of the reversal distances (i.e., the minimal number of reversals needed to transform one gene order to another) to the three input gene orders is minimal. For inferring phylogenetic trees based on gene orders solving this reversal median problem (RMP) is a basic operation used in several algorithms ([12,17,18,10]). It is known that certain gene groups are preserved during evolution. Since it is difficult to determine functionally what a gene group is it has been proposed to consider common combinatorial structures between gene orders as gene groups. Conserved intervals

⋆ This work was supported by the German Research Foundation (DFG) through the project "Deep Metazoan Phylogeny" within SPP 1174.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 305–316, 2007.

([6]), or — as used in this paper — common intervals ([21,16]) are considered in evolutionary scenarios ([3,8] and [1,9]). Unfortunately, even computing the perfect reversal distance, i.e. the minimum number of reversals that preserve all common intervals between two given gene orders is an NP-complete problem ([14]). Without the restriction that common intervals should be preserved the problem (and computing a corresponding minimum length sequence of reversals) is polynomial time solvable ([15,5] and [20]). But finding a median for three given gene orders is NP-hard ([13]). There are exact ([13,19]) and heuristic ([12,10,7]) median solvers available. In the inspiring paper of Bérad *et al.* ([2]), it was shown that parsimonious preserving sorting scenarios can be computed in polynomial time for many instances. This results are based on the strong interval tree, which is a tree data structure for representing the set of common intervals of a set of gene orders. The strong interval tree can be computed in linear time for a constant number of input gene orders. In this paper we use strong interval trees for the computation of perfect reversal median scenarios.

Basic definitions are given in Section 2. How the perfect RMP is solved with algorithm TCIP is described in Section 3. Empirical results on biological mtDNA data and on random data are presented in Section 4. Conclusions are given in Section 5.

2 Basic Definitions

A *permutation of size* n is a permutation of the elements in $\{1,2,\ldots,n\}$. A *signed* permutation of size n is a permutation of size n where every element has an additional sign ("+" or "−") that defines its orientation ("+" is usually omitted). A *reversal* $\rho(i,j)$, $1 \le i \le j \le n$ applied to a signed permutation π of size n transforms it into $\pi \circ \rho = (\pi_1,\ldots,\pi_{i-1},-\pi_j,\ldots,-\pi_i,\pi_{j+1},\ldots,\pi_n)$. A sorting scenario for two signed permutations π and σ is a sequence of reversals $\rho_1,\ldots,\rho_d$ that transforms π into σ. If d is minimal the sequence is called parsimonious and d is the *reversal distance* $d(\pi,\sigma)$ between π and σ.

An interval of a permutation π is a set of consecutive elements of the permutation π. Let Π be a set of signed permutations of size n. A *common interval* ([21,16]) of Π is a subset of $\{1,2,\ldots,n\}$ that is an interval in all $\pi \in \Pi$. Each singleton and the set of all elements are called *trivial common intervals*. Let $C(\Pi)$ be the set of all common intervals of Π. Two common intervals c and c' *overlap* if $c \cap c' \neq \emptyset$, $c \not\subset c'$, and $c' \not\subset c$. If two intervals do not overlap they *commute*. A common interval is called a *strong common interval*, if it does not overlap with any other common interval. The set of all strong common intervals can be computed in $O(kn)$ for k signed permutations of length n [4]. A *strong interval tree* is a tree of the strong common intervals of Π. The root node is the interval containing all elements and the leaves are the singletons, inner nodes and edges of inner nodes are defined by the minimal inclusion relation of the intervals. A reversal is said to be *preserving* if it does not destroy any common interval (i.e., a reversal is not preserving, if there exists a common interval, such that it does not exists after applying the reversal). The *perfect reversal distance* $d_p(\pi,\sigma)$ between two signed permutations π and σ is the minimum number of preserving reversals necessary to transform π into σ.

The reversal median problem (RMP) is to find for given signed permutations $\Pi = \{\pi^1,\pi^2,\ldots,\pi^k\}$, a signed permutation μ which has a minimal sum of reversal distances to the given signed permutations, i.e., the score $\sum_{i=1}^{k} d(\pi^i, \mu)$ has to be minimal. A *median scenario* is defined as the reversal sequences from each of the input permutations to the median, such that the total number of reversals is minimal. A sequence of permutations that is traversed from an input permutation π^i towards the median is called the *i-th trace*. For the perfect RMP (pRMP), similar to the definition of the perfect distance measure, only preserving reversals are allowed, i.e., no reversal is allowed to break one of the common intervals of the three signed input permutations, and for the median μ it holds $C(\Pi) = C(\Pi \cup \{\mu\})$. Note the slight difference to the problem as defined in [9], where for the computation of the perfect distance only the pairwise common intervals of the input permutations and the median was taken into account (and not the common intervals of all three input sequences). A modification of the RMP is the oriented RMP (oRMP) which is to find a median such that for a given tuple of signs $(s^1, s^2, \ldots, s^k)$ the value $\sum_{i=1}^{k} d(\pi^i, s^i \circ \mu)$ has to be minimal (if $s = +$, then $s \circ \pi = \pi$, if $s = -$, then $s \circ \pi = -\pi$, i.e., the complete signed permutation is inverted).

3 Solving the Perfect Reversal Median Problem

In this section a generalization of strong interval trees as introduced in [2] for pairs of permutations is defined. The structure of this generalized strong interval tree strongly influences the amount of computation time needed to solve the pRMP. Methods are presented for solving the pRMP based on the different properties of this tree. These methods are used in the proposed algorithm for the pRMP.

3.1 Definitions and The Median Parity Theorem

Similar to [2] we define quotient permutations as follows: Let π be a permutation. A partition $\mathscr{I} = \{I_1, \ldots, I_k\}$ of the elements of π into common intervals is a congruence relation. The *quotient permutation* associated with $\mathscr{I}$ denoted $P_{|\mathscr{I}}$ is defined as follows: (i precedes j in $P_{|\mathscr{I}}$) iff (I_i precedes I_j). A quotient permutation $P_{|\mathscr{I}}$ defined by the children of a node in the strong interval tree of a permutation is *increasing* (resp. *decreasing*) *linear*, if $P_{|\mathscr{I}}$ is the identity (resp. the inverse of the identity). Otherwise a quotient permutation is *prime*. Using the definition of quotient permutations, the property linear or prime is assigned to the nodes of strong interval trees as follows. When using input permutations $\pi_1, \ldots, \pi_k$ in each node of the strong interval tree $k-1$ quotient permutations (relative to π_1) are induced. Each quotient permutation is either prime or linear (increasing or decreasing). If all $k-1$ quotient permutations are linear, the node is said to be linear. Let $\Pi = \{\pi^1, \ldots, \pi^k\}$ be the given input permutations. W.l.o.g. we rename all elements in the signed permutations such that π^1 is the identity permutation. A tuple $s = (s^1, \ldots, s^k)$, where $s^i \in \{+, -\}$, is called a k-sign. The inverted tuple of s, i.e. the tuple for which each sign is inverted, is denoted as $\overline{s}$. A k-signed strong interval tree $T_S^k(\Pi)$ assigns k-signs to the nodes of the interval tree, such that

i.) each leaf gets the k-sign according to the signs of the corresponding elements of the input permutations,
ii.) the l-th entry in the k-sign of a linear node is $+$ (resp. $-$) if the quotient permutation of the node in π^l (relative to π^1) is increasing (resp. decreasing),
iii.) a prime node inherits the sign of the parent if the parent is linear.

For a node I its k-sign is denoted $s(I)$. A k-signed strong interval tree is called *unambiguous*, if the parent of every prime node is linear, and *ambiguous* otherwise. Examples for 3-signed strong interval trees are given in the following subsections.

From the following propositions the first one is a slightly modified version from [2]. The other two are easy to see and the technical proofs are omitted.

Proposition 1. *A (median) scenario is preserving iff each of the reversals is either a node of the strong interval tree or the union of children of a prime node of the strong interval tree.*

Proposition 2. *Let Π be the set of input permutations of a pRMP, μ be a median of Π, and S be a preserving reversal median scenario for transforming the permutations in Π to μ. Then on each trace the elements that correspond to the children of a prime node are reordered by S, such that they are all in the same order at the end of the trace. Furthermore, the reversals in S change the k-signs of linear and prime nodes, such that for each node I in the tree all signs of its k-sign are either positive or negative, i.e., either $\forall i : s^i(I) = +$ or $\forall i : s^i(I) = -$.*

Proposition 3. *Let I be a node in a strong interval tree. Let $\rho^i_1, \ldots, \rho^i_m$ be the sequence of m reversals applied on the i-th trace. For all reversals, for which $\rho_j \cap I = \emptyset$ or $\rho_j \subset I$ holds, the k-sign $s(I)$ of node I is not changed. Each reversal with $I \subseteq \rho_j$ inverts the i-th sign of $s(I)$.*

Theorem 1. *(The Median Parity Theorem) Let I be a node of the strong interval tree $T^3_s(\Pi)$ of signed permutations $\Pi = \{\pi^1, \pi^2, \pi^3\}$. If node I has a linear parent J, with $s(I) \neq s(J)$ and $s(I) \neq \overline{s}(J)$, then there exists a reversal of I on one of the traces, which leads to either $s(I) = s(J)$ or $s(I) = \overline{s}(J)$. This reversal belongs to any perfect median scenario of Π.*

Proof. Let S be a perfect median scenario and let I be a node in the strong interval tree for which $s(I) \neq s(J)$ and $s(I) \neq \overline{s}(J)$ holds and an inversion of I does not occur in the perfect scenario. By Proposition 1 and 3 $s(I) \neq s(J)$ and $s(I) \neq \overline{s}(J)$ still holds after applying all reversals from S, as a reversal that inverts a sign in $s(I)$ also inverts the corresponding sign in $s(J)$. Therefore, $s(I) = s(J)$ or $s(I) = \overline{s}(J)$ is not achievable by S. Thus by Proposition 2 S can not be a perfect median scenario. From this contradiction follows immediately that a reversal of I must occur in any perfect scenario. Because $s(I) = s(J)$ or $s(I) = \overline{s}(J)$ can always be achieved by applying the reversal of I on exactly one trace only, this reversal has to occur in any perfect (parsimonious) median scenario. (Note, that the possibility of applying the reversal of I on two traces would not be parsimonious.) □

The Median Parity Theorem can be easily generalized to more than three permutations. For an even number of permutations in Π the set of reversals to be applied on the traces may be not unique. In the case of an odd number of input permutation there is always a unique set of reversals.

In the following we explain how the pRMP can be solved in the case of strong interval trees that are i) unambiguous and have no prime node, ii) unambiguous and have prime nodes, and iii) ambiguous. Our algorithm *TCIP* (Tree Common Interval Preserving) uses all the methods that are explained in the following subsections. Due to space limitations a presentation of the pseudo code is omitted.

3.2 Unambiguous Trees Without Prime Nodes

If no prime node occurs in the strong interval tree a perfect median scenario and the only existing median is directly defined by the tree. Suppose the 3-signs of a node I and its parent are $s = (s^1, s^2, s^3)$, respectively $s_p = (s_p^1, s_p^2, s_p^3)$. As the signs of each node have to become equal Theorem 1 implies that:

1. if the number of sign differences is 1, i.e. $s^2 \neq s_p^2$ or $s^3 \neq s_p^3$, then the corresponding (second or third) trace is extended by inverting I. This leads to $s = s_p$.
2. if the number of sign differences is 2, i.e. $s^2 \neq s_p^2$ and $s^3 \neq s_p^3$, then the first trace is extended by inverting I. This leads to $s = \overline{s}_p$.

Note, that three sign differences can not occur as $s^1 = +$ for all linear nodes, because π^1 is the identity permutation. We illustrate the case of no prime nodes with a small example.

Example 1. Let $\pi^1 = (1\ 2\ 3)$, $\pi^2 = (1\ \overline{2}\ \overline{3})$, and $\pi^3 = (\overline{3}\ 1\ \overline{2})$ be the three input permutations ($\overline{i}$ is an abbreviation for $-i$). The only non-trivial strong common interval for these permutations is $C = \{\{1,2\}\}$. The 3-signed strong interval tree is depicted in Figure 1(a). There are two differences between the signs of node $\{2\}$ and its parent ($s(\{2\}) = (+,-,-)$ and $s(\{1,2\}) = (+,+,+)$). This induces the reversal of $\{2\}$ in π^1. The nodes $\{1,2\}$ and $\{3\}$ have one different sign compared to their parents, leading to the reversals of $\{1,2\}$ in π^3 and the reversal of $\{3\}$ in π^2. The root node has signs $(+,+,-)$, so π^3 has to be inverted completely. The traces of the parsimonious median scenario with these 4 reversals and the median $(1\ \overline{2}\ 3)$ are: $\pi^1 \curvearrowright (1\ \overline{2}\ 3)$, $\pi^2 \curvearrowright (1\ \overline{2}\ 3)$, and $\pi^3 \curvearrowright (\overline{3}\ 2\ \overline{1}) \curvearrowright (1\ \overline{2}\ 3)$ where $\curvearrowright$ indicates a reversal operation.

Note, that the perfect median scenario as well as the median can be computed in linear time for strong interval trees that are unambiguous and have no prime node.

3.3 Unambiguous Trees with Prime Nodes

In this subsection we assume that each prime node has a linear parent. The basic idea is to compute for each prime node a parsimonious scenario for the median problem that is induced by the three quotient permutations of the prime node. As a prime node with a linear parent inherits the k-sign from its parent, it is clear which instance of the oRMP has to be solved, i.e., the sign vector for the oRMP is known. Note, that the oRMP for a given sign vector $(s^1, \ldots, s^k)$ and signed permutations $\pi^1, \ldots, \pi^k$, i.e., minimizing $\sum_{i=1}^{k} d(\pi^i, s^i \circ \mu)$, can be solved as a standard median problem where $\sum_{i=1}^{k} d(s^i \circ \pi^i, \mu)$ is minimized. Hence, an oRMP can be solved with a standard RMP solver, e.g., Caprara's exact median solver [13].

Example 2. Let $\pi^1 = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10)$, $\pi^2 = (1\ 2\ 3\ 4\ \overline{6}\ 9\ 7\ 5\ \overline{8}\ 10)$, and $\pi^3 = (1\ 2\ \overline{9}\ \overline{8}\ 7\ \overline{6}\ \overline{5}\ \overline{4}\ \overline{3}\ \overline{10})$ be the three input permutations. The non-trivial strong common intervals for $\{\pi^1, \pi^2, \pi^3\}$ are $C = \{\{3,4,5,6,7,8,9\}, \{5,6,7,8,9\}\}$. The 3-signed strong interval tree is depicted in Figure 1(b). The only prime node inherits the 3-sign $(+,+,-)$ from the linear parent, and the three quotient permutations are $\Gamma^1 = (1\ 2\ 3\ 4\ 5)$, $\Gamma^2 = (\overline{2}\ 5\ 3\ 1\ \overline{4})$, and $\Gamma^3 = (\overline{5}\ \overline{4}\ 3\ \overline{2}\ \overline{1})$. Solving the oRMP for 3-sign (+,+,-) — or equivalently solving the RMP for Γ^1, Γ^2, and $-\Gamma^3$ — leads to the only median $\mu_p = (1\ 2\ \overline{3}\ 4\ 5)$ with a subtree

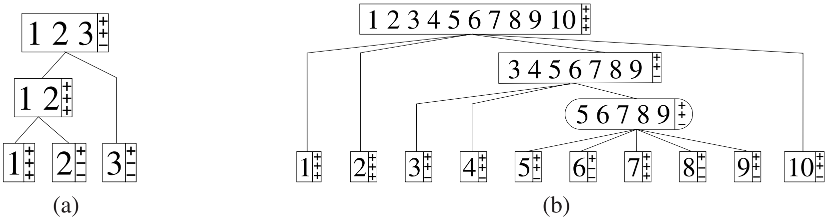


Fig. 1. Unambiguous strong interval trees; tree for Example 1 in the text (no prime nodes, left); tree for Example 2 in the text (one prime node, right); linear nodes are depicted without rounded corners, prime nodes with round corners; signs indicate the 3-sign of a node

median score of 4. Note, that in general there may be more than one median, but all of them have the same score. One parsimonious scenario for this oRMP is defined by the following traces: $\Gamma^1 \overset{3}{\curvearrowright} \mu_p$, $\Gamma^2 \overset{531}{\curvearrowright} (\overline{2}\ \overline{1}\ \overline{3}\ \overline{5}\ \overline{4}) \overset{\overline{21}}{\curvearrowright} (1\ 2\ \overline{3}\ \overline{5}\ \overline{4}) \overset{\overline{54}}{\curvearrowright} \mu_p$, and $\Gamma^3 = -\mu_p$. The reversals in the oRMP scenario directly correspond to reversals in the pRMP scenario. The overall perfect reversal scenario leading to $\mu = (1\ 2\ 3\ 4\ 5\ 6\ \overline{7}\ 8\ 9\ 10)$ is then defined by the following traces: i) $\pi^1 \overset{7}{\curvearrowright} \mu$ (corresponding to $\Gamma^1 \overset{3}{\curvearrowright} \mu_p$ in the oRMP), ii) $\pi^2 \overset{\text{3 reversals}}{\curvearrowright} \mu$ (derived from the 3 reversals of $\Gamma^2 \curvearrowright \mu_p$ in the oRMP), and iii) the reversals induced by the sign differences of linear nodes: $\pi^3 \overset{\overline{10}}{\curvearrowright} (1\ 2\ \overline{9}\ \overline{8}\ 7\ \overline{6}\ \overline{5}\ \overline{4}\ \overline{3}\ 10) \overset{\overline{9}...\overline{3}}{\curvearrowright} \mu$. The perfect median scenario has a score of 6.

Note, that solving the pRMP with separate prime nodes can be accomplished by solving for each prime node one RMP. Although there may be several prime nodes, the number of elements per prime node is usually much smaller than the original permutation length. This may lead to a computation time reduction compared to solving the original RMP.

3.4 Ambiguous Trees

Similar to perfect sorting scenarios, the most difficult case for the pRMP occurs when there exist connected prime nodes in the strong interval tree. The induced separate connected prime node subtrees can be solved separately. In the following we first explain a brute force approach and suggest some improvements afterwards. For each prime node with a prime node parent the 3-sign is unknown. Hence, we have to assume each possible 3-sign for such prime nodes. The 3-sign of a prime node defines the signs of the corresponding elements in the parent node. Therefore we have to solve i) all instances of the oRMP for the prime child nodes and ii) all possible oRMPs in the parent node, where the signs of the elements are changed according to the signs of the corresponding prime children. I.e. each possible combination of sign assignments for the corresponding elements in the parent node has to considered. This leads to a number of different oRMP instances to be solved, that is growing exponentially with the number of prime node children of a prime node. Note, that the different oRMP instances can be solved for each prime node independently. After all instances are solved, we have to assume each possible 3-sign assignment combinations for all prime nodes without a 3-sign and use the pre-computed median solutions in order to determine the minimal score for a prime node subtree. The root node is an exception as the root may only have the 3-sign $(+,+,+)$ or $(-,-,-)$.

The minimal number of reversals induced by a prime node subtree can not be computed for each prime node separately, but all possible combinations of 3-sign assignments to the prime nodes of the subtree have to be taken into account. The reason is the following: Let P' be a prime node with a prime node parent P. An optimal 3-sign assignment for P' (i.e., a 3-sign for P' that leads to a minimal score for the solution of the oRMPs) induces a certain sign assignment for the elements in the parent node P. An exact oRMP solution for P with the element signs that are defined by the exact oRMP solution of P' may now lead to a score, that is larger than the score that would have been achieved when using other sign assignments for the corresponding element in P. Therefore, the minimal score for such a prime node component can only be computed by iterating over all possible sign assignments.

The number of oRMP instances that have to be solved in a prime node is 8^e, where e is the number of edges to other prime nodes: each of the 8 possible 3-signs has to be considered, and all of the 8 sign assignments corresponding to the $e-1$ prime node children have to be considered. This number can be reduced easily to 4^e, as solving an oRMP with three reverted input permutations leads to the reversed median (in the root only the 3-sign $(+,+,+)$ has to be considered). The 8 possible sign assignments for the elements can be easily reduced to 4 by a simple renaming procedure. After all oRMPs are solved, all possible combinations of 3-sign assignments have to be considered for each prime node subtree. Hence, the overall number of combinations to be considered for one subtree is 4^{v-1}, where v is the number of nodes in the subtree. For subtrees with a small degree for the nodes usually the possible combinations of 3-sign assignments is the limiting factor, whereas for subtrees where the node degree(s) is/are large, the number of oRMPs to be solved is the limiting factor. Note again, that computing the score for one combination takes only constant time, whereas solving an oRMP instance is done with an exponential computation time algorithm (but is usually very fast to compute for short gene orders, e.g., for mtDNAs). The case of solving the pRMP for ambiguous trees is illustrated in the following example:

Example 3. Let $\pi^1 = (1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9\ 10)$, $\pi^2 = (2\ 1\ \overline{10}\ \overline{9}\ \overline{6}\ 4\ 5\ \overline{3}\ 8\ 7)$, and $\pi^3 = (1\ \overline{10}\ \overline{9}\ \overline{2}\ \overline{5}\ \overline{4}\ \overline{3}\ 8\ 7\ 6)$ be the three input permutations. The non-trivial strong common intervals for these gene orders are $C=\{\{3,4,5,6,7,8\},\{3,4,5\},\{4,5\},\{7,8\},\{9,10\}\}$. There are two prime nodes in this tree, namely $P_1 = \{3,4,5,6,7,8\}$, and the root node $P_2 = \{1,\ldots,10\}$. The 3-signed strong interval tree is depicted in Figure 2(a). The three quotient permutations (corresponding to π^1,π^2, and π^3) of prime node P_1 are $(1\ 2\ 3)$, $(\overline{2}\ \overline{1}\ \overline{3})$, and $(\overline{1}\ \overline{3}\ 2)$. The three quotient permutations of prime node P_2 are $(1\ 2\ \pm 3\ 4)$, $(2\ 1\ \overline{4}\ \pm 3)$, and $(1\ \overline{4}\ \overline{2}\ \pm 3)$. For each of the possible 3-sign assignments of P_1 the RMP is solved, the score $s(p_1)$ and the medians are given in Table 1. The assumed 3-sign for P_1 defines the sign of element 3 in P_2. The solutions of the RMP of node P_2 when element 3 is signed according to the 3-sign of P_1 are also given in Table 1. In the case of this small example each 3-sign for P_1 leads to an overall score $s(P_1,P_2) = 9$ for the subtree induced by the prime nodes (Note that in general different scores can be achieved for different 3-sign combinations). The computation of one median by applying the permutations implied by oRMP solutions and reversals resulting from sign

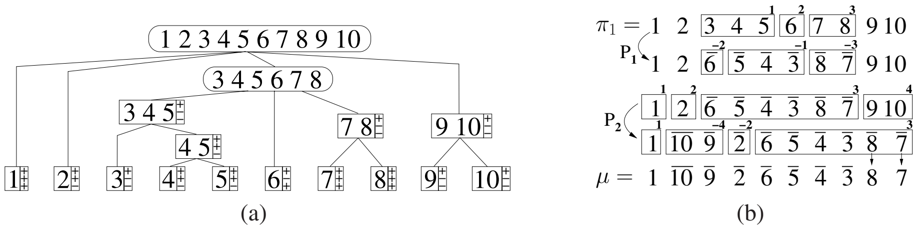


Fig. 2. The ambiguous strong interval tree for Example 3 in the text (left); computation of the perfect median beginning with π^1 (also for Example 3); from top to bottom: the permutations induced by the oRMP medians of prime nodes P_1 and P_2 are applied, and finally two reversals due to sign differences of the linear nodes {7} and {8} are applied, resulting in a median μ; small numbers above the boxes represent the indices of the corresponding quotient permutation of the prime nodes (resp. the prime node median)

Table 1. Median computation for prime nodes P_1 and P_2 in Example 3; given are the 3-sign assignments of P_1 (first column), the median score $s(P_1)$ in node P_1, the median score $s(P_2)$ when signs for element 3 are chosen corresponding to the 3-sign in the first column, and the overall score for prime nodes P_1 and P_2

3-sign of P_1	$s(P_1)$	medians (P_1)	$s(P_2)$	medians (P_2)	$s(P_1,P_2)$
(+,+,+)	4	$\{\ (\overline{2}\ \overline{1}\ \overline{3})\ \}$	5	$\{\ (1\ \overline{4}\ \overline{2}\ 3)\ \}$	9
(+,+,-)	3	$\{\ (\overline{2}\ \overline{1}\ \overline{3})\ \}$	6	$\{\ (1\ \overline{4}\ \overline{2}\ \overline{3}), (1\ \overline{4}\ \overline{2}\ 3), (1\ \overline{4}\ 3\ \overline{2}), (2\ 1\ \overline{4}\ \overline{3})\ \}$	9
(+,-,+)	4	$\{\ (\overline{1}\ \overline{3}\ 2), (3\ 1\ 2)\}$	5	$\{\ (1\ \overline{4}\ \overline{3}\ \overline{2})\ \}$	9
(+,-,-)	4	$\{\ (\overline{2}\ \overline{1}\ \overline{3})\ \}$	5	$\{\ (1\ \overline{4}\ \overline{2}\ \overline{3}), (2\ 1\ \overline{4}\ \overline{3})\ \}$	9

differences between linear nodes starting from π^1 is shown in Figure 2(b). Note that the traces can be easily computed with known algorithms (e.g. [20]). The median for the pRMP was determined by solving 4 oRMPs of size 3 for P_1 and 4 oRMPs of length 4 for P_2. 4 combinations of 3-sign assignments had to be tested (comp. Table 1). The interval tree has 3 linear nodes with parents having a different 3-sign ({7}, {8}, and {4,5}), therefore the overall score is 12. There are 5 exact solutions of this pRMP, which can be found when all medians of P_1 and P_2 are combined.

When computing evolutionary scenarios the input permutations might be circular (i.e. each circular shift of a gene order is assumed to be equal) or linear (not circular). Furthermore the permutations can be directed, (i.e., $\pi \neq -\pi$) or undirected (i.e., $\pi = -\pi$). So far we only considered the linear directed case. The circular undirected case can be derived easily from the linear directed case as follows: Let π^1, π^2, and π^3 be three circular undirected gene orders with a perfect median scenario S. Then, by applying inversion and circular shift, there exist three gene orders $\varpi^i = \pi^i, 1 \leq i \leq 3$, such that the first element of $\varpi^i, 1 \leq i \leq 3$ is 1. Note, that in a circular scenario we can replace each reversal by its circular complement without changing the score or the resulting median. Hence, for $\varpi^i = \pi^i, 1 \leq i \leq 3$ there exists a (circular undirected) perfect scenario S' such that no reversal has a start index larger than its end index and such that the score of S equals the score of S'. As element 1 never has to change its position in this scenario, S'

is also a perfect median scenario under the assumption that $\varpi^i, 1 \leq i \leq 3$ are linear and directed. Therefore, we can solve the circular undirected case by shifting and inverting the input permutations as explained.

For linear undirected input permutations the handling of the root node has to be changed: i.) linear root nodes can be left unchanged, since the relative orientation of the three permutations does not matter, ii.) for prime root nodes all possible 3-sign assignments have to be tested (instead of only $(+,+,+)$) and the assignments which lead to a minimal score have to be chosen.

4 Results

As test instances for the pRMP random test data sets as well as biological data sets have been used. The random test instances were generated as follows. Starting with the identity permutation d random reversals were applied to generate a random permutation. Three permutations generate a triple instance.

As biological data the mitochondrial genome orders from [11] which were marked as complete are used. From this data set all gene orders were removed which did not have the standard set of 37 mitochondrial genes (13 protein coding-, 2 rRNA-, and 22 tRNA- genes). With the help of taxonomy information in the mitochondrial database and manual inspection of the phylogenetic tree given from the NCBI the species were grouped into taxa to get subsets of reasonable size. The following groups of Deuterostomia were used: The Chordata (Cho) and their sub groups Actinopterygii (Act), and Sarcopterygii (Sar), as well as the remaining Deuterostomia Hyperotreti, Cephalochordata, Echinodermata, and Hemichordata (HCEH). In the Protostomia we have grouped the Arthropoda (Art) and their sub groups Crustacea (Cru), Hexapoda (Hex), and the remaining Protostomia Annelida, Pogonophora, Brachiopoda, and Mollusca (APBM). Finally, the Nematoda and the Platyhelminthes (NP) form a group. Note, that nearly all mitochondrial gene orders of the species in the last group have no gene atp8, hence here we have used a reduced set of 36 genes. The group of all species (All) excludes the NP group. The number of unique gene orders for each group can be found in Table 2 (#). For the analysis all $\binom{n}{3}$ possible triples of the groups were used.

4.1 Properties of Strong Interval Trees

In this subsection properties of strong interval trees, which are crucial for the runtime of the pRMP solver, will be given. For the random data sets 1000 test instances have been created for each of the following combinations of sequence length $n = 100$ and number of random reversals d applied to the identity: $d \in \{1, 2, 5, 10, 25\}$. In Table 2 (resp. Table 3) results are presented for the biological (resp. random) data sets. For random as well as biological data sets there are many instances which have only linear nodes. For the three Chordate groups (Act, Sar, Cho) the majority of the interval trees have no prime nodes (59-74%), also in the Hexapoda there are many instances which have only linear nodes (29%). For random data sets only triples generated with few reversals have no prime nodes. Usually the trees have only a very small number of prime nodes. The maximal number of prime nodes in all test runs is 4 and the average is always ≤ 1.12. When more than one prime node occurs, they are usually separated and the size of the

Table 2. Properties of strong common interval trees for the mtDNA data set; #: number of unique gene orders; p: percentage of triple instances for which the tree has at least one prime node; q: number of prime nodes; c: number of prime node subtrees; r: number of oRMPs to be solved; t: number of 3-sign combinations to be tested; l: maximal number of children of the prime nodes; for columns $q, \ldots, l$ only instances are taken into account, for which at least one prime node exists; if two values are given in a column, the first is the average over all instances and the second value (in parentheses) is the maximal value

	#	p	q	c	r	t	l
Act	27	0.26	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	11.93 (22)
Sar	28	0.41	1.06 (2)	1.06 (2)	1.06 (8)	1.06 (4)	4.43 (10)
Cho	47	0.40	1.03 (2)	1.03 (2)	1.03 (8)	1.03 (4)	7.44 (23)
HCEH	11	0.99	1.10 (2)	1.07 (2)	1.33 (8)	1.18 (4)	23.87 (30)
Cru	18	0.91	1.09 (2)	1.02 (2)	1.51 (8)	1.23 (4)	20.26 (35)
Hex	15	0.71	1.08 (3)	1.07 (3)	1.10 (8)	1.08 (4)	15.88 (32)
Art	42	0.92	1.12 (4)	1.09 (3)	1.28 (24)	1.17 (16)	19.14 (35)
APBM	16	1.00	1.01 (2)	1.01 (2)	1.02 (8)	1.01 (4)	32.35 (36)
NP	9	0.99	1.05 (3)	1.05 (3)	1.05 (3)	1.05 (3)	32.65 (35)
All	115	0.95	1.04 (4)	1.03 (3)	1.07 (24)	1.05 (16)	26.73 (36)

Table 3. Properties of strong common interval trees for random data sets; notation see Table 2; d: number of reversals applied for generating data set; length of the permutations: $n = 100$

d	p	q	c	r	t	l
1	0.67	1.00 (1)	1.00 (1)	1.00 (1)	1.00 (1)	3.76 (5)
2	0.98	1.10 (3)	1.09 (3)	1.18 (8)	1.13 (4)	7.53 (11)
5	1.00	1.05 (3)	1.04 (3)	1.14 (24)	1.09 (16)	22.63 (29)
10	1.00	1.01 (2)	1.01 (2)	1.03 (8)	1.02 (4)	42.96 (51)
25	1.00	1.00 (2)	1.00 (2)	1.03 (8)	1.01 (4)	76.44 (88)

subtrees induced by the prime nodes is small. Therefore the number of oRMPs to be solved and the number of combinations that have to be tested is small. The worst case in terms of the number of oRMPs to be solved and combinations to be tested is the case of 3 prime nodes which are all in one induced subtree (this occurred 8 times in the All data set). In this case 24 oRMPs had to be solved and 16 combinations had to be tested. Furthermore for gene orders with relative small evolutionary distances (e.g. the Chordates or the Arthropods in the biological data set or small values for d in the random data set) the size of the solved oRMPs is small compared to the original RMP (i.e. the length of the signed permutations). When $d \leq 10$ was used the maximal length of a prime node over all instances was 51. Summarizing, the strong interval tree analysis indicates that pRMP medians seems to be computable with reasonable computation time.

4.2 Computing RMP Medians

In this subsection we present computation times for computing all exact pRMP medians. For the random data sets 100 test instances have been created for each of the following combinations of sequence lengths n and number of random reversals d applied to the identity: $(n,d) \in \{(10,1),\ldots,(10,5),(30,1),\ldots,(30,10),(50,2),(50,4),\ldots,$

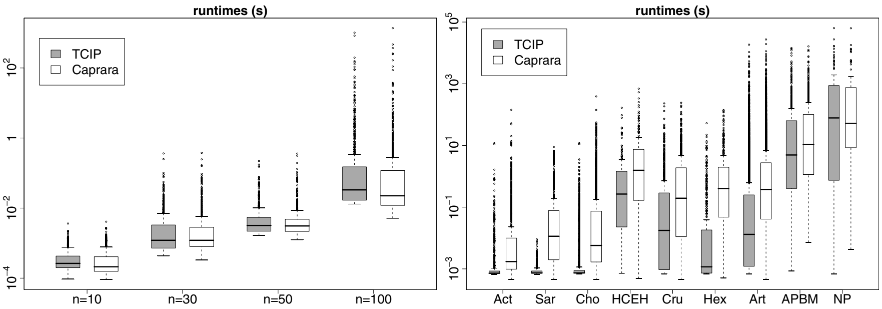


Fig. 3. Comparison of computation times: TCIP solving pRMP, Caprara's algorithm solving RMP (values given in seconds); depicted are the boxplots for random data sets of length $n \in \{10, 30, 50, 100\}$ (left) and different biological data sets (right)

$(50,12),(100,5),(100,10),\ldots,(100,30)\}$. In Figure 3 boxplots are given for the computation times for the perfect median computations of i) the different subgroups for the mtDNA data sets, and ii) the random data sets with sequence length $n \in \{10, 30, 50, 100\}$. All test runs were done on PCs with AMD Opteron 2.0 GHz processors.

It can be clearly seen, that computing all pRMP medians with TCIP is usually faster then computing all RMP medians with Caprara's median solver. On average TCIP was 25.43 times faster on the biological data sets, for the Sarcopterygii data set TCIP was even 260.82 times faster compared to Caprara's RMP solver (which is known to be very fast compared to other approaches). The only exception is the NP data set where Caprara's median solver is about 1.25 times faster. Note again, that Caprara's median solver does not solve the pRMP but the RMP. Nevertheless, we compared the computation times to show the good performance of TCIP. In [9] an algorithm called ECIP was presented, which also solves the pRMP. ECIP was compared to a simply modified version of Caprara's median solver and shown to be faster than the latter. As ECIP and Caprara's modified median solver were not able to solve all problem instances presented here in reasonable time, we have not included their computation times. Algorithm TCIP is much faster than ECIP, often achieving speedups of $10^2 - 10^3$ for the instances presented in this paper. We verified the correctness also by comparing the results of all algorithms. Algorithm TCIP is freely available from the authors.

5 Conclusion

Based on strong interval trees we introduced a new algorithm called TCIP for solving the perfect reversal median problem (pRMP), i.e., no reversal in the median scenario is allowed to break one of the common intervals of the three signed input permutations. It was shown that the hardness of the problem strongly depends on the structure of the strong interval tree. Solving a pRMP instance is usually accomplished by solving several smaller RMP instances. For data sets of mitochondrial gene orders and for randomly generated data sets it was shown, that perfect scenarios can be computed even faster than standard scenarios, although even computing the perfect distance is an NP-complete problem. In our future work we will include TCIP in algorithm amGRP [10], a state-of-the-art algorithm for computing phylogenetic trees based on frequently solving the RMP.

References

1. S. Bérard, A. Bergeron, and C. Chauve. Conservation of combinatorial structures in evolution scenarios. In *Comparative Genomics, RECOMB 2004 International Workshop, RCG 2004*, number 3388 in LNBI, pages 1–15, 2004.
2. S. Bérard, A. Bergeron, C. Chauve, and C. Paul. Perfect sorting by reversals is not always difficult. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4(1):4–16, 2007.
3. A. Bergeron, M. Blanchette, A. Chateau, and C. Chauve. Reconstructing ancestral gene orders using conserved intervals. In *Proc. WABI*, number 3240 in LNCS, pages 14–25, 2004.
4. A. Bergeron, C. Chauve, F. de Montgolfier, and M.Raffinot. Computing common intervals of K permutations, with applications to modular decomposition of graphs. In *Proc. ESA*, number 3669 in LNCS, pages 779–790. Springer-Verlag, Berlin, 2005.
5. A. Bergeron, J. Mixtacki, and J. Stoye. Reversal distance without hurdles and fortresses. In *Proc. CPM*, number 3109 in LNCS, pages 388–399. Springer Verlag, 2004.
6. A. Bergeron and J. Stoye. On the similarity of sets of permutations and its applications to genome comparison. *J. Comp. Biol.*, 13(7):1345–1354, 2006.
7. M. Bernt, D. Merkle, and M. Middendorf. A parallel algorithm for solving the reversal median problem. In *Proc. Parallel Processing and Applied Mathematics - Bio-Computing Workshop (PBC'5)*, number 3911 in LNCS, pages 1089–1096, 2005.
8. M. Bernt, D. Merkle, and M. Middendorf. Genome rearrangement based on reversals that preserve conserved intervals. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 3(3):275–288, 2006.
9. M. Bernt, D. Merkle, and M. Middendorf. The reversal median problem, common intervals, and mitochondrial gene orders. In *Computational Life Sciences II - Proc. 2nd International Symposium CompLife*, number LNCS in 4216, pages 52–63, 2006.
10. M. Bernt, D. Merkle, and M. Middendorf. Using median sets for inferring phylogenetic trees. *Bioinformatics*, 23(2):e129–e135, 2007.
11. J.L. Boore. Mitochondrial gene arrangement database, 2006. http://evogen.jgi.doe.gov/.
12. G. Bourque and P. A. Pevzner. Genome-scale evolution: Reconstructing gene orders in the ancestral species. *Genome Res.*, 12(1):26–36, 2002.
13. A. Caprara. The reversal median problem. *INFORMS Journal on Computing*, 15(1):93–113, 2003.
14. M. Figeac and J. Varré. Sorting by reversals with common intervals. In *Proc. WABI*, number 3240 in LNBI, pages 26–37, 2004.
15. S. Hannenhalli and P. Pevzner. Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 178–189. ACM Press, 1995.
16. S. Heber and J. Stoye. Finding all common intervals of k permutations. In *Proc. CPM*, number 2089 in LNCS, pages 207–218, 2001.
17. B. Moret, A. Siepel, J. Tang, and T. Liu. Inversion medians outperform breakpoint medians in phylogeny reconstruction from gene-order data. In *Proc. WABI*, number LNCS in 2452, pages 521–536, 2002.
18. B. Moret, J. Tang, and T. Warnow. Reconstructing phylogenies from gene-content and gene-order data. *Mathematics of Evolution and Phylogeny*, 2005.
19. A. Siepel and B. Moret. Finding an optimal inversion median: Experimental results. In *Proc. WABI*, number 2149 in LNCS, pages 189–203, 2001.
20. E. Tannier, A. Bergeron, and M.-F. Sagot. Advances in sorting by reversals. *accepted at Discrete Applied Mathematics*, 2005.
21. T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 2(26):290 – 309, 2000.

Genomic Signatures from DNA Word Graphs

Lenwood S. Heath and Amrita Pati

Department of Computer Science, Virginia Tech, Blacksburg, VA 24061-0106
{heath,apati}@vt.edu

Abstract. Genomes have both deterministic and random aspects, with the underlying DNA sequences exhibiting features at numerous scales, from codons and *cis*-elements through genes and on to regions of conserved or divergent gene order. The DNA Words program aims to identify mathematical structures that characterize genomes at multiple scales. The focus of this work is the fine structure of genomic sequences, the manner in which short nucleotide sequences fit together to comprise the genome as an abstract sequence, within a graph-theoretic setting. A DNA word graph is a generalization of a de Bruijn graph that records the occurrence counts of node and edges in a genomic sequence. A DNA word graph can be derived from a genomic sequence generated by a finite Markov chain or a subsequence of a sequenced genome. Both theoretically and empirically, DNA word graphs give rise to genomic signatures. Several genomic signatures are derived from the structure of a DNA word graph, including an information-rich and visually appealing genomic bar code. Application of genomic signatures to several genomes demonstrate their practical value in identifying and distinguishing genomic sequences.

1 Introduction

The genome $\mathcal{G}$ of an organism is a set of long nucleotide sequences modeled, within a formal language framework, as strings over $\Sigma_{\mathrm{DNA}} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$, the DNA alphabet. While $\mathcal{G}$ itself is a unique mathematical structure for the organism, a genome is typically quite large (e.g., billions of bases) and differs slightly from one individual of a species to another. Fix a genomic sequence H that is a substring of some string in $\mathcal{G}$. Intuitively, a genomic signature for an organism is a mathematical structure $\theta(H)$ derived from H, which, ideally, can be efficiently computed, is significantly smaller to represent than H, and, if H is sufficiently representative of $\mathcal{G}$, can uniquely identify the original organism. The intent is that the signature of other large substrings from $\mathcal{G}$ be highly similar to $\theta(H)$ and distinguishable from signatures of other organisms. A genomic signature is judged along two, typically antagonistic, dimensions: (1) the amount of compression achieved by $\theta(H)$, and (2) its effectiveness in identifying the genome.

Karlin and Burge [1] were among the first to use the term *genomic signature.* They define the *dinucleotide odds ratio* or *relative abundance*, which is the 16 functions defined for dinucleotides XY by

$$\rho_{XY}(H) = \frac{f_{XY}(H)}{f_X(H) f_Y(H)},$$

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 317–328, 2007.

where $f_x(H)$ is the frequency of string x as a substring in H. They observe that ρ values are similar throughout a genome and may reflect the net response of the genome to selection pressures. They compare dinucleotide odds ratios for a number of organisms and demonstrate their capability of distinguishing organisms. Karlin et al. [2] observe that dinucleotide odds ratios typically range from 0.78 to 1.23. They define the *delta distance* between strings x and y to be

$$\delta(x, y) = \frac{1000}{16} \sum_{\substack{\text{dinucleotide} \\ XY}} |\rho_{XY}(x) - \rho_{XY}(y)|.$$

Jernigan and Baran [3] demonstrate that the delta distance between strings sampled from a genome is preserved over a wide range of string lengths $|x|$ and $|y|$. For bacterial species, Coenye and Vandamme [4] correlate delta distance with 16S rDNA sequence similarity and DNA-DNA hybridization values. They find a strong negative correlation between δ and 16S rDNA similarity among groups of species with low δ and high 16S rDNA similarity. They also demonstrate an overall high negative correlation between δ and DNA-DNA hybridization values. Deschavanne et al. [5] construct images from oligonucleotide frequencies. For 57 prokaryotic genomes, Sandberg et al. [6] compare G+C content, oligonucleotide frequency, and codon bias. Dufraigne et al. [7] and van Passel et al. [8] employ oligonucleotide frequencies to identify regions of horizontal gene transfer (HGT) in prokaryotes. Carbone et al. [9] correlate the ecological niches of 80 Eubacteria and 16 Archaea to codon bias used as a genomic signature.

As part of our DNA Words program investigating mathematical invariants derived from genomes, we examine the finest scale in graph-theoretic terms, employing a generalization of de Bruijn graphs. One frequently exploited observation is that a string over Σ_{DNA} defines a walk in a suitably defined de Bruijn graph. Closely related is the correspondence of such a string to an Eulerian tour in a suitably defined multigraph. Applications include DNA physical mapping, DNA sequence assembly, and multiple sequence alignment problems [10,11,12,13,14]. In Section 2, we formalize the mathematical basis for graph-theoretic genomic signatures and, in Section 3, prove that, under reasonable probabilistic assumptions, these signatures characterize a genomic sequence with high probability and distinguish it from those of other organisms. We present empirical studies that support the theoretical results in Section 4 and conclude in Section 5.

2 Preliminaries

An *alphabet* is a finite, non-empty set of symbols; the *DNA alphabet* is $\Sigma_{\mathrm{DNA}} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. A *string* or *word* x over Σ_{DNA} is a finite sequence $x = \sigma_1\sigma_2 \cdots \sigma_w$ of symbols from Σ_{DNA}; its *length* $|x|$ is w. A single chromosome in a genome is typically written as the string of nucleotides on one DNA strand. A *genomic sequence* is a chromosomal sequence or any substring of it. $\mathcal{G}$ is the set of all chromosomal sequences from an organism. Nucleotide frequencies vary among organisms, while, as Fickett et al. [15] observe, the frequencies of A's and T's (and hence of G's and C's) are approximately constant within a single genome. If x and y are strings, then $\mathrm{occ}\,(x, y)$ is the count of occurrences of x in y.

Fix a word length $w \geq 1$. Let $l = 4^w$. The *order-w state space* is $\mathcal{S}^w = \Sigma_{\mathrm{DNA}}^w$, the set consisting of the l words of length w. The *order-w de Bruijn graph* $\mathcal{DB}^w = (\mathcal{S}^w, E)$ is the directed graph, where $(x_i, x_j) \in E$ when $x_i\sigma = \iota x_j$, for some $\sigma, \iota \in \Sigma_{\mathrm{DNA}}$; such an edge is labeled σ [16]. Fig. 1 illustrates the order-2 de Bruijn graph.

Let $H \in \Sigma_{\mathrm{DNA}}^*$ have length $|H| = n$; we think of H as a long genomic sequence that traces a walk in $\mathcal{DB}^w$. The *vertex count* of x_i in H is $\mathrm{vc}\,(x_i, H) = \mathrm{occ}\,(x_i, H)$, while the *edge count* of edge $(x_i, x_j) \in E$ in H, where $x_i\sigma = \gamma x_j$, is $\mathrm{ec}\,((x_i, x_j), H) = \mathrm{occ}\,(x_i\sigma, H)$. The *order-$w$ DNA word graph* $\mathcal{DNA}^w\,(H)$ is $\mathcal{DB}^w$ together with labels $\mathrm{vc}\,(x_i, H)$ for each $x_i \in \mathcal{S}^w$ and $\mathrm{ec}\,((x_i, x_j), H)$ for each $(x_i, x_j) \in E$. For $x_i, x_j \in \mathcal{S}^w$, the *frequency of x_j after x_i in H* is

$$\mathrm{Freq}\,((x_i, x_j), H) = \begin{cases} 0 & \text{if } (x_i, x_j) \notin E \text{ or } \mathrm{vc}\,(x_i, H) = 0; \\ \dfrac{\mathrm{ec}\,((x_i, x_j), H)}{\mathrm{vc}\,(x_i, H)} & \text{otherwise.} \end{cases}$$

For $1 \leq i \leq l$, let x_i be the i^{th} element of $\mathcal{S}^w$ in lexicographic order. The *order-w word count vector* χ_H^w of H is the l-vector having components $\mathrm{occ}\,(x_i, H)$, in lexicographic order.

We consider Markov chains with state space $\mathcal{S}^w$ and having nonzero transition probabilities only for edges in $\mathcal{DB}^w$; such a Markov chain is called an *order-w de Bruijn chain (DBC)*. Let $\mathcal{DC}$ be an order-w DBC with $l \times l$ transition probability matrix $P = (p_{ij})$; here, p_{ij} is the probability of a one-step transition from state x_i to state x_j [17]. P is sparse, with at most 4 nonzero entries per row. The *order-w DBC $\mathcal{DC}^w(H)$ for genomic sequence H* has transition probabilities $p_{ij} = \mathrm{Freq}\,((x_i, x_j), H)$. Experimental results suggest that genomic sequences are sufficiently large and diverse in their composition to sample all words in $\mathcal{S}^w$ for reasonably small $w \in [1, 5]$. Hence, the DBCs generating such sequences are irreducible. Genomic sequences are also adequately random to assume that the DBCs generating them are aperiodic and recurrent non-null. Throughout, we assume that all DBC are ergodic and hence that there is a unique *stationary distribution* $\pi = (\pi_i)$ on $\mathcal{S}^w$ satisfying $\pi P = \pi$ [17]. This assumption does not hold for a genomic sequence that consists of systematic repeats of a small subset of words from $\mathcal{S}^w$.

For a genome $\mathcal{G}$ and a genomic sequence H taken from $\mathcal{G}$, a *genomic signature* for H is a function θ that maps H to a mathematical structure $\theta(H)$. Ideally, $\theta(H)$ is able to identify sufficiently large substrings that come from $\mathcal{G}$ and to distinguish H from genomic sequences of other genomes. To be useful, $\theta(H)$ must be efficiently computable. Of course, a representation of $\mathcal{G}$ itself satisfies the requirements, but offers no advantage in space.

Fixing word length $w \geq 1$, we obtain $\mathcal{DNA}^w\,(H)$, with associated $\mathrm{vc}\,(x_i, H)$ and $\mathrm{ec}\,((x_i, x_j), H)$. We define several candidate signatures. The simplest is the *vertex count vector* $\theta^{cv}(w) = (\mathrm{vc}\,(x_i, H))_{i=1}^{l}$, requiring space $\Theta(4^w \lg n)$.

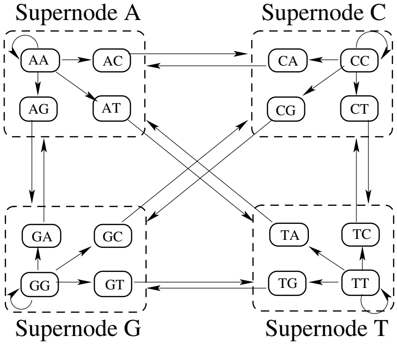


Fig. 1. Representation of the de Bruijn graph $\mathcal{DB}^2$ in terms of supernodes and superedges. Each supernode consists of the 4 nodes with the same 1-symbol prefix in their labels and is closed by a dotted boundary. An edge from a node to a supernode represents a set of edges from the node to all nodes in the supernode. For example, the edge from node AC to supernode C represents the set of edges $\{(\mathrm{AC}, \mathrm{CA}), (\mathrm{AC}, \mathrm{CC}), (\mathrm{AC}, \mathrm{CG}), (\mathrm{AC}, \mathrm{CT})\}$.

Additional signatures come from interplay between the graph structure $\mathcal{DB}^w$ and the count vectors. Let $\psi \geq 0$ be an integer *threshold*. Let $E^{\leq \psi} = \{(i,j) \in E \mid \mathrm{ec}((i,j), H) \leq \psi\}$, be the set of edges with counts at most ψ. Then *edge deletion* is the process of deleting edges in $E^{\leq \psi}$ from $\mathcal{DB}^w$, while varying ψ from 0 to $\Xi = \max\{\mathrm{ec}((i,j), H) \mid (i,j) \in E\}$ and deleting edges with tied counts in arbitrary order. The *ψ-edge deletion of $\mathcal{DB}^w$* is $\mathcal{DB}^w(\psi) = (\mathcal{S}^w, E - E^{\leq \psi})$. As ψ increases from 0 to Ξ, the number of connected components in $\mathcal{DB}^w(\psi)$ increases from 1 to l, while the number of isolated vertices increases from 0 to l. The *vertex deletion order θ^{vdo}* is the permutation of $\mathcal{S}^w$ giving the order in which vertices become isolated during edge deletion. Let ψ_i be the smallest integer such that $\mathcal{DB}^w(\psi_i)$ has precisely i connected components. The *component-based edge deletion vector θ^{ced}* is the l-vector whose i^{th} component is the number of edge deletions required to go from $i-1$ to i components. The *vertex-based edge deletion vector θ^{ved}* is the l-vector whose i^{th} component is the number of edge deletions required to go from $i-1$ to i isolated vertices. The *ordered vertex-based edge deletion vector θ^{oed}* is the l-vector whose i^{th} component is the total number of edge deletions required to isolate the vertex x_i, where x_i is the i^{th} element of $\mathcal{S}^w$ in lexicographic order. For two vector-based signatures θ_1 and θ_2, let $d(\theta_1, \theta_2)$, be the L_1 metric in l-dimensional real space.

3 Theory and Methods

We imagine every biological sequence to be generated by a formal model that can be approximated by a DBC. In this section, we build a theoretical framework to analyze distances between genomic signatures in terms of the parameters of the DBC generating them.

Let $\mathcal{DC}$ be an ergodic, order-w DBC. Let H be a sequence generated by $\mathcal{DC}$, where $|H| = n$. If $x_i, x_j \in \mathcal{S}^w$, the probability of transition from state x_i to state x_j is given by $p_{i,j}$, and the stationary probability for x_i is π_i.

Let $x = \sigma_1\sigma_2 \ldots \sigma_w \in \Sigma^w_{\text{DNA}}$. A *period* of x is an integer i, where $1 \leq i \leq w$, such that $x[1 \ldots i] = x[w - i + 1 \ldots w]$. Two occurrences $H[i \ldots i + w - 1]$ and $H[j \ldots j + w - 1]$ of x in H *overlap* if $i \leq j \leq i + w - 1$ or $j \leq i \leq j + w - 1$. An *x-clump* in H is a maximal subsequence of one or more consecutive overlapping occurrences of x. For example, 2 is a period of $x =$ AACAA, and AACAACAACAACAA is a clump with 4 occurrences of x. Waterman [18] notes that the count of a rare DNA word in H is a function of the number of x-clumps in H, which approximately follows a Poisson distribution [18], with parameter λ_β (derived below). Let x be a DNA word with shortest period d. Then a *declumping* event with respect to x is defined as the event of not observing the string $x' = x[1 \ldots d]$. Suppose the probability of occurrence of x' is p_x. Then the probability of a declumping event is given by $q_x = 1 - p_x$. The number of occurrences of x within a clump is approximately geometric with mean $1/p_x$ [18].

Lemma 1. *Let X_x be the random variable that is the number of occurrences of word x in genomic sequence H. Then the probability generating function of X_x is*

$$f_{X_x}(t) = \exp\left(\frac{\lambda_x(t-1)(1-p_x)}{1-q_x t}\right).$$

Proof. Let Z be the random variable that is the number of x-clumps in H, and let C_i be the number of occurrences of x in the i^{th} clump. Hence,

$$X_x = \sum_{i=1}^{Z} C_i.$$

Since Z has (approximately) a Poisson distribution with parameter λ_x, the probability generating function for Z is

$$f_Z(t) = \sum_{k=0}^{\infty} e^{-\lambda_x} \frac{(\lambda_x t)^k}{k!} = e^{\lambda_x(t-1)}.$$

The probability generating function for each C_i is

$$f_C(t) = p_x \sum_{k=0}^{\infty} (q_x t)^k = \frac{p_x}{1-q_x t}.$$

Hence the probability generating function for X_x is

$$f_{X_x}(t) = f_Z(f_C(t)) = \exp\left(\lambda_x\left(\frac{p_x}{1-q_x t} - 1\right)\right) = \exp\left(\frac{\lambda_x(t-1)(1-p)}{1-q_x t}\right). \square$$

Lemma 2. $\mathbf{E}[X_x] = \dfrac{\lambda_x q_x}{p_x}$ *and* $\text{Var}[X_x] = \dfrac{\lambda_x q_x}{p_x}\left(2 + \dfrac{q_x}{p_x}\right)$.

Proof. By results in [17], $\mathbf{E}[X_x] = f'_{X_x}(1)$ and $\mathrm{Var}[X_x] = f''_{X_x}(1) + f'_{X_x}(1) - (f'_{X_x}(1))^2$. The lemma follows by calculation. □

Lemma 3. *Let H be a genomic sequence of length n, and let χ_H^w be its word count vector. Fix threshold $\tau > 0$. Then*

$$\Pr\left[d\left(\chi, \mathbf{E}\left[\chi\right]\right) \geq l\tau\right] \leq \prod_{x \in \mathcal{S}^w} \frac{n\pi_x}{\tau^2}\left(2 + \frac{q_x}{p_x}\right).$$

Proof. Let $\chi_H^w = (X_1, X_2, \ldots X_l)$. Since $\mathbf{E}[X_x] = n\pi_x = (\lambda_x q_x)/p_x$, we have $\lambda_x = (n\pi_x p_x)/q_x$. The distance between χ and $\mathbf{E}[\chi]$ is $d(\chi, \mathbf{E}[\chi]) = \sum_{x \in \mathcal{S}^w} |X_x - \mathbf{E}[X_x]|$. By Chebyshev's bound and Lemma 2, we obtain

$$\Pr\left[|X_x - \mathbf{E}\left[X_x\right]| \geq \tau\right] \leq \frac{\mathrm{Var}\left[X_x\right]}{\tau^2} = \frac{\lambda_x q_x}{p_x \tau^2}\left(2 + \frac{q_x}{p_x}\right) = \frac{n\pi_x}{\tau^2}\left(2 + \frac{q_x}{p_x}\right).$$

The lemma follows from the resulting inequality:

$$\Pr\left[d\left(\chi, \mathbf{E}\left[\chi\right]\right) \geq l\tau\right] \leq \prod_{x \in \mathcal{S}^w} \Pr\left[|x - \mathbf{E}\left[x\right]| \geq \tau\right].$$ □

Theorems 1 and 2 address the ability of word count vectors to identify and distinguish DBCs.

Theorem 1. *Let $\mathcal{DC}$ be an order s DBC. Let H_1 and H_2 be two genomic sequences of length n generated independently by $\mathcal{DC}$. Let χ_1 and χ_2 be their respective order-w word count vectors. Then,*

$$\Pr\left[d\left(\chi_1, \chi_2\right)\right) \geq 2l\tau\sqrt{n}\right] \leq \frac{2e}{\tau^{2l}}.$$

Proof. The component-wise expected values in χ_1 and χ_2 are the same. Their expected difference is therefore the 0 vector. Therefore,

$$d\left(\chi_1 - \mathbf{E}\left[\chi_1\right], \chi_2 - \mathbf{E}\left[\chi_2\right]\right) = d\left(\chi_1, \chi_2\right).$$

Furthermore using $T = \sqrt{n}\tau$ we obtain,

$$\Pr\left[d\left(\chi_1, \mathbf{E}\left[\chi_1\right]\right) \geq lT\right] = \Pr\left[d\left(\chi_2, \mathbf{E}\left[\chi_2\right]\right) \geq lT\right].$$

Using the above equations and Lemma 3, we obtain

$$\Pr\left[d\left(\chi_1 - \mathbf{E}\left[\chi_1\right], \chi_2 - \mathbf{E}\left[\chi_2\right]\right) \geq 2lT\right] = \Pr\left[d\left(\chi_1, \chi_2\right) \geq 2lT\right].$$

$$\begin{aligned}\Pr\left[d\left(\chi_1, \chi_2\right) \geq 2lT\right] &\leq \Pr\left[d\left(\chi_1, \mathbf{E}\left[\chi_1\right]\right) \geq lT\right] + \Pr\left[d\left(\chi_2, \mathbf{E}\left[\chi_2\right]\right) \geq lT\right] \\ &= 2\prod_{x \in \mathcal{S}^w} \frac{n\pi_x}{T^2}\left(2 + \frac{q_x}{p_x}\right).\end{aligned}$$

If $x' = x[1 \dots d]$, where d is the smallest period of x, $|x| \geq |x'|$. Therefore, $p_x \geq \pi_x$ and $\frac{q_x}{p_x} \leq \frac{1-\pi_x}{\pi_x}$, which yields

$$\Pr\left[d\left(\chi_1, \chi_2\right) \geq 2lT\right] \leq \frac{2}{\tau^{2l}} \prod_{x \in \mathcal{S}^w} (1+\pi_x).$$

From the Arithmetic-Geometric Inequality [19], we obtain

$$\prod_{x \in \mathcal{S}^w} (1+\pi_x) \leq \left(\frac{\sum_{x \in \mathcal{S}^w} (1+\pi_x)}{l}\right)^l = \left(1+\frac{1}{l}\right)^l$$

From the above results we have

$$\Pr\left[d\left(\chi_1, \chi_2\right) \geq 2l\tau\sqrt{n}\right] \leq \frac{2}{\tau^{2l}} \left(1+\frac{1}{l}\right)^l.$$

As $l \to \infty$ the theorem follows. □

Let H_1 and H_2 be genomic sequences of length n, generated independently by DBCs $\mathcal{DC}_1$ and $\mathcal{DC}_2$ of orders s_1 and s_2, respectively. Let $\chi_1 = \chi_1^{H_1}$ and $\chi_1 = \chi_2^{H_2}$ be their order-w word count vectors. This assumption formalizes the separation of genomic sequences obtained from different organisms.

Assumption 1. *There exists a non-negative real number* $\gamma \in (0, 1]$ *such that*

$$\Pr\left[d\left(\mathbf{E}\left[\chi_1\right], \mathbf{E}\left[\chi_2\right]\right) \geq 3l\tau\sqrt{n}\right] \geq \gamma.$$

Then, the distance $d\left(\chi_1, \chi_2\right)$ can distinguish $\mathcal{DC}_1$ and $\mathcal{DC}_2$.

Theorem 2. *let $X_{x,1}$ and $X_{x,2}$ denote the counts of x in H_1 and H_2, respectively. Assuming that H_1 and H_2 are both generated by Markov chains $\mathcal{DC}_1'$ and $\mathcal{DC}_2'$ of order w, let $\pi_{x,1}$ and $\pi_{x,2}$ denote the stationary probabilities of state x in $\mathcal{DC}_1'$ and $\mathcal{DC}_2'$, respectively. If there exists a constant γ as in Assumption 1 then,*

$$\Pr\left[d\left(\chi_1, \chi_2\right) \geq l\tau\sqrt{n}\right] \geq \gamma - \prod_{x \in \mathcal{S}^w} \frac{1}{\tau^2}(2\pi_{x,1}+1) - \prod_{x \in \mathcal{S}^w} \frac{1}{\tau^2}(2\pi_{x,2}+1).$$

Proof. Treating $d\left(\chi_1, \chi_2\right)$, $d\left(\chi_1, \mathbf{E}\left[\chi_1\right]\right)$, $d\left(\chi_2, \mathbf{E}\left[\chi_2\right]\right)$, and $d\left(\mathbf{E}\left[\chi_1\right], \mathbf{E}\left[\chi_2\right]\right)$ as distances d, d_1, d_2, and d_3, respectively, in 1-dimensional space and using $T = \tau\sqrt{n}$ we obtain,

$$d_3 \leq d + d_1 + d_2$$

$$\Pr\left[d_3 \geq 3lT\right] \leq \Pr\left[d \geq lT\right] + \Pr\left[d_1 \geq lT\right] + \Pr\left[d_2 \geq lT\right].$$

From Assumption 1, Lemma 3, and $\pi_x \leq p_x$ we obtain,

$$\gamma \leq \Pr\left[d\left(\chi_1, \chi_2\right) \geq lT\right] + \prod_{x \in \mathcal{S}^w} \frac{n\pi_{x,1}}{T^2}\left(2+\frac{q_{x,1}}{p_{x,1}}\right) + \prod_{x \in \mathcal{S}^w} \frac{n\pi_{x,2}}{T^2}\left(2+\frac{q_{x,2}}{p_{x,2}}\right),$$

$$\Pr\left[d\left(\chi_1, \chi_2\right) \geq l\tau\sqrt{n}\right] \geq \gamma - \prod_{x \in \mathcal{S}^w} \frac{1}{\tau^2}(2\pi_{x,1}+1) - \prod_{x \in \mathcal{S}^w} \frac{1}{\tau^2}(2\pi_{x,2}+1).$$

The theorem follows. □

By Theorem 2, the probability that the distance between the word count vectors of sequences generated by different DBCs exceeds $l\tau\sqrt{n}$, increases with τ. Sequences assumed to be generated by two different DBC with sufficiently different stationary distributions would have a high probability of being separated by a large distance.

4 Results and Discussion

To evaluate our genomic signatures, we employed chromosomal sequences from the following organisms: *Arabidopsis thaliana* (AT), *Borrelia burgdorferi* (BB), *Caenorhabditis elegans* (CE), *Chlamydophila pneumoniae* (CP), *Chlamydia muridarum* (CM), *Escherichia coli* (EC), *Homo sapiens* (HS), and *Saccharomyces cerevisiae*(SC).

We computed the vertex count vector $\theta^{cv}(3)$, the vertex deletion order vector $\theta^{vdo}(3)$, the vertex-based edge deletion vector $\theta^{ved}(3)$, the component-based edge deletion vector $\theta^{ced}(3)$, and the ordered vertex-based edge deletion vector $\theta^{oed}(3)$ signatures for SC chromosomes 4,5, and 8, CE chromosomes 1,3, and 4, and AT chromosomes 1,2, and 3, followed by all pairwise Pearson correlation coefficients between signature vectors for each type of signature. Figure 2(a) illustrates that despite Theorem 2, even the nucleotide compositional differences between diverse species such as SC, CE, and AT are not captured by the traditional method of using multi(tri, here)-nucleotide frequencies. Correlation coefficients between $\theta^{cv}(3)$ signatures of chromosomes of the same species are very close to those between $\theta^{cv}(3)$ signatures of chromosomes of different species. Nucleotide frequencies for dimers, tetramers, and pentamers display similar characteristics (results not shown). For the same set of genomic sequences, the $\theta^{ved}(3)$ and $\theta^{ced}(3)$ signatures display much greater discriminatory power (Fig. 2) than the $\theta^{vdo}(3)$ and $\theta^{cv}(3)$ signatures.

To test the efficiency of the θ^{ced} and θ^{ved} signatures in identifying target genomes from smaller unknown genomic sequences, we randomly sampled 1 Mb sequences from the existing genomic sequences. We computed signatures from these sequences and matched them to existing genomic signatures. In most cases, the signature derived from a random sequence sampled from a genomic sequence of an organism O displayed highest positive correlation with genomic signatures derived from chromosomes of O. However, this behavior was not conserved across all samples. Varying the word length at which the signatures were computed did not alter this behavior significantly. To identify the organism from which a sequence originates more accurately, we combined the properties of θ^{vdo} and θ^{ved} to compute the ordered vertex-based edge deletion vector θ^{oed}. The θ^{oed} signature precisely predicts the organism corresponding to a short DNA sequence (1 Mb) using a database of previously computed θ^{oed}'s for various organisms. Empirical results suggest that θ^{oed} performs best at order 5 for bacterial genomes.

To demonstrate the efficiency of the θ^{oed} signature we tested its performance using 5 chromosomal sequences from 4 species of the *Rhizobiaceae* family. The species with their Entrez refseq numbers are: *Agrobacterium tumefaciens str.*

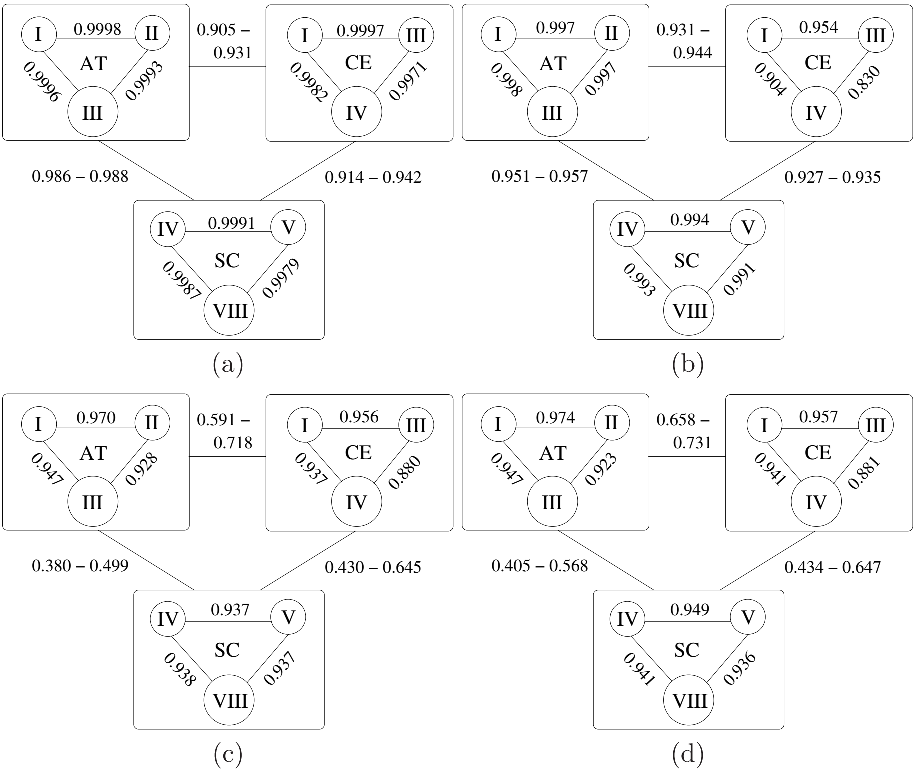
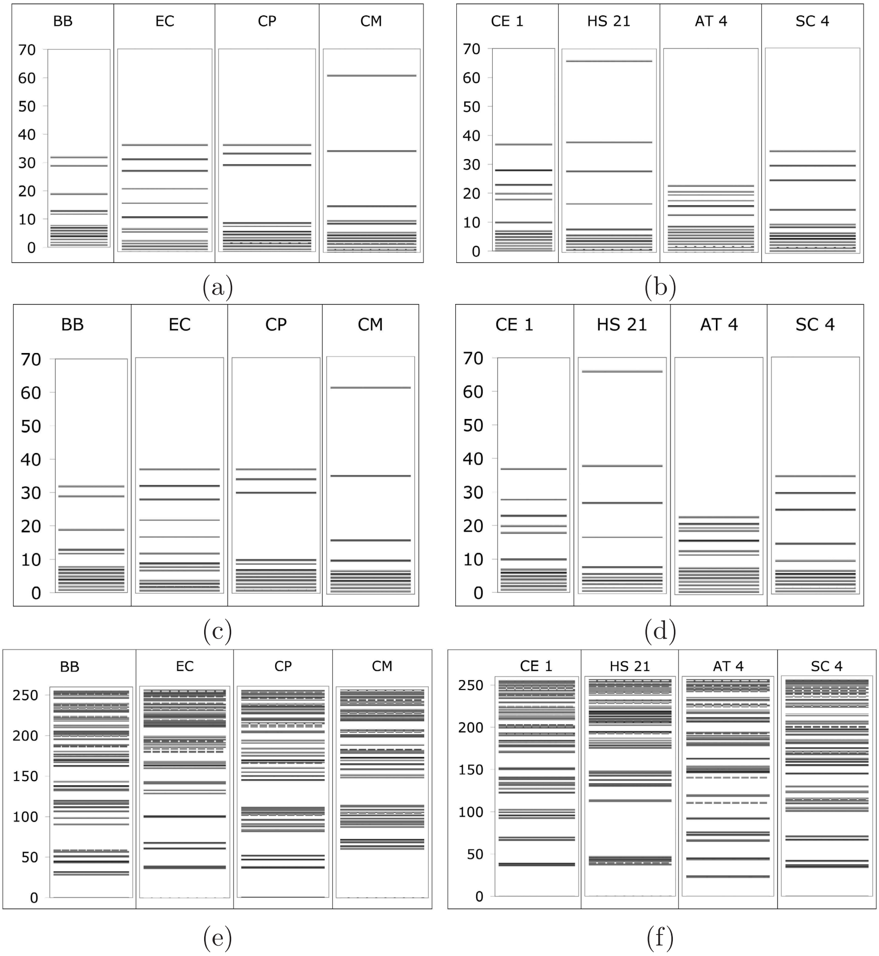


Fig. 2. Correlation graphs of Pearson correlation coefficients between order-3 (a) θ^{cv}s, (b) θ^{vdo}s, (c) θ^{ved}s, (d) θ^{ced}s. Text in the circles indicates chromosome numbers. Edges between any two enclosed subgraphs are labeled with the range of correlations on all 9 edges between the subgraphs. p-values of all the correlation coefficients given are below 10^{-3}. Lengths of the chromosomes used above are as follows: AT I: 30 M, AT II: 19 M, AT III: 15 M, CE I: 15 M, CE III: 14 M, CE IV: 17 M, SC IV: 1.5 M, SC V: 0.5 M, SC VIII: 0.5 M.

C58 (2 chromosomes, `NC_003062` and `NC_003063`), *Rhizobium etli CFN 42* (`NC_007761`), *Rhizobium leguminosarum bv. viciae 3841* (`NC_008380`), and *Sinorhizobium meliloti 1021* (`NC_003047`). 1 Mb segments were randomly sampled from each of these sequences, their θ^{oed}s were computed, and matched to the database of θ^{oed}s. At order 5, all random segements correctly matched up to their respective target genomes demonstrating that the θ^{oed} signature is sensitive enough to differentiate between species within a family. The efficiencies of identification of correct matches were 60%, 80%, and 100% at orders 3, 4, and 5, respectively. The θ^{oed} signature is computable in $O(n + 4^{w+1} \log(4^{w+1}))$ time, where n is the input sequence length and w is the order at which the signature is computed.

Figure 3 illustrates $\theta^{ved}(3)$ and $\theta^{ced}(3)$ signatures for the four eukaryotes and the four prokaryotes. Although $\theta^{ved}(3)$ and $\theta^{ced}(3)$ appear similar, one cannot



(a) (b)

(c) (d)

(e) (f)

Fig. 3. $\theta^{ved}(3)$s for (a) four prokaryotes and (b) four eukaryotes. $\theta^{ced}(3)$s for (c) four prokaryotes and (d) four eukaryotes. $\theta^{oed}(3)$s for (e) four prokaryotes and (f) four eukaryotes. Numbers denote chromosomes. The above is a grayscale representation. Colors play a vital role in these signatures. Each shaded-bar represents a specific component in the signature. Figures (e) and (f) illustrate that the $\theta^{oed}(3)$ bar code of each species is unique and sufficiently different from the $\theta^{oed}(3)$ bar codes of other species.

conclude that an increase of one in the number of isolated vertices precisely coincides with an increase of one in the number of connected components during edge deletion. As the DNA word graph fragments, in the early stage it is natural that the number of components grows at a rate greater than or equal to the number of isolated vertices. However, in the later stages, the graph is sufficiently fragmented so that an increase in the number of connected components of the graph coincides with the isolation of a vertex.

5 Conclusion

The genomic signatures introduced in this paper are systematically derived from the structure of DNA word graphs obtained from genomic sequences. Moreover, distances between such signatures can be characterized within a probabilistic framework in terms of the parameters of the underlying DBC assumed to generate the sequences. For each organism, both eukaryotic and prokaryotic, it is possible to derive a θ^{oed}-bar code from a sufficiently long genomic sequence of that organism that uniquely identifies the organism among competing genomic sequences. When sufficient sequence for an organism is present in a biological sample, the target organism for the sample can be retrieved by querying an already existing database of signatures. All order-w signatures discussed in this paper are compact and computable in $\Theta(4^w \lg n)$ time and space. The amount of sequence needed to create an order-w signature representative of its species is exponential in w.

In practice, DNA sequences that need to be matched to a target organism can be much smaller than 1 Mb. We have found that each genomic sequence has a separate set of specifications for order and sample sequence size for best results (work in progress). We continue to investigate bounds on the minimum amount of sequence required to achieve an effective θ^{oed} signature and on alternate signatures for high order w that sample counts for a few length-w strings instead of requiring counts for all 4^w strings.

Acknowledgments

We thank two anonymous reviewers for their useful comments and suggestions.

References

1. Karlin, S., Burge, C.: Dinucleotide relative abundance extremes — A genomic signature. Trends in Genetics **11**(7) (1995) 283–290
2. Karlin, S., Mrazek, J., Campbell, A.M.: Compositional biases of bacterial genomes and evolutionary implications. Journal of Bacteriology **179**(12) (1997) 3899–913
3. Jernigan, R.W., Baran, R.H.: Pervasive properties of the genomic signature. BMC Genomics **3** (2002) 9 pages
4. Coenye, T., Vandamme, P.: Use of the genomic signature in bacterial classification and identification. Systematic and Applied Microbiology **27**(2) (2004) 175–185
5. Deschavanne, P.J., Giron, A., Vilain, J., Fagot, G., Fertil, B.: Genomic signature: Characterization and classification of species assessed by chaos game representation of sequences. Molecular Biology and Evolution **16**(10) (1999) 1391–1399
6. Sandberg, R., Branden, C.I., Ernberg, I., Coster, J.: Quantifying the species-specificity in genomics signatures, synonymous codon choice, amino acid usage, and G+C content. Gene **311** (2003) 35–42
7. Dufraigne, C., Fertil, B., Lespinats, S., Giron, A., Deschavanne, P.: Detection and characterization of horizontal transfers in prokaryotes using genomic signature. Nucleic Acids Research **33**(1) (2005) 12 pages

8. van Passel, M.W.J., Bart, A., Thygesen, H.H., Luyf, A.C.M., van Kampen, A.H.C., van der Ende, A.: An acquisition account of genomic islands based on genome signature comparisons. BMC Genomics **6** (2005) 10 pages
9. Carbone, A., Kepes, F., Zinovyev, A.: Codon bias signatures, organization of micro-organisms in codon space, and lifestyle. Molecular Biology and Evolution **22**(3) (2005) 547–561
10. Pevzner, P.A.: DNA physical mapping and alternating Eulerian cycles in colored graphs. Algorithmica **13**(1-2) (1995) 77–105
11. Pevzner, P.A., Tang, H.X., Waterman, M.S.: An Eulerian path approach to DNA fragment assembly. Proceedings of The National Academy of Sciences of The United States Of America **98**(17) (2001) 9748–9753
12. Zhang, Y., Waterman, M.S.: An Eulerian path approach to global multiple alignment for DNA sequences. Journal of Computational Biology **10**(6) (2003) 803–819
13. Raphael, B., Zhi, D.G., Tang, H.X., Pevzner, P.: A novel method for multiple alignment of sequences with repeated and shuffled elements. Genome Research **14**(11) (2004) 2336–2346
14. Zhang, Y., Waterman, M.S.: An Eulerian path approach to local multiple alignment for DNA sequences. Proceedings of The National Academy of Sciences of The United States Of America **102**(5) (2005) 1285–1290
15. Fickett, J.W., Torney, D.C., Wolf, D.R.: Base compositional structure of genomes. Genomics **13**(4) (1992) 1056–1064
16. Rosenberg, A.L., Heath, L.S.: Graph separators, with applications. Frontiers of Computer Science. Kluwer Academic/Plenum Publishers (2000)
17. Feller, W.: An Introduction to Probability Theory and Its Applications. Third edn. Volume I. John Wiley & Sons Inc., New York (1968)
18. Waterman, M.: Introduction to Computational Biology. First edn. Academic Press Inc., Boston, MA (1995)
19. Cauchy, A.L.: Cours d'analyse de l'École Royale Polytechnique. Première partie. Instrumenta Rationis. Sources for the History of Logic in the Modern Age, VII. Cooperativa Libraria Universitaria Editrice Bologna, Bologna (1992) Analyse algébrique. [Algebraic analysis], Reprint of the 1821 edition, Edited and with an introduction by Umberto Bottazzini.

Enhancing Motif Refinement by Incorporating Comparative Genomics Data

Erliang Zeng and Giri Narasimhan

Bioinformatics Research Group (BioRG), School of Computing and Information Sciences,
Florida International University, Miami, Florida, 33199, USA
{ezeng001,giri}@cis.fiu.edu

Abstract. Transcription factor binding sites (TFBS) are often located in the upstream regions of genes and transcription factors (TFs) cause transcription regulation by binding at these locations. Predicting these binding sites is a difficult problem, and traditional methods have a high degree of false positives in their predictions. Comparative genomics data can help to improve motif predictions. In this paper, a new strategy is presented, which refines motif by taking the comparative genomics data into account. Tested with the help of both simulation data and biological data, we show that our method makes improved predictions. We also propose a new metric to score a motif profile. This score is biologically motivated and helps the algorithm in its predictions.

1 Introduction

Transcription, the process to produce messenger RNA (mRNA) from DNA, is a key step in the "Central Dogma" of life. Transcription regulation is a complex process and a lot of proteins are involved. A protein that regulates transcription by binding to short DNA sequences located in the upstream region of a gene is called a transcription factor (TF); the short DNA sequences are referred as transcription factor binding sites (TFBS) or regulatory elements. Typically, a single TF regulates a large set of genes. The transcription regulation of a single gene may be the effect of a single TF or the combined effect of multiple TFs. The upstream region of each gene regulated by the same TF must have at least one binding site specific to that particular TF. All the binding sites for the same TF share a high similarity but need not be identical. The bioinformatics problem is to find these sites and to describe them in an efficient way.

A *motif* is a sequence signature that provides a description of a TFBS. The possible ways to represent a motif include a consensus sequence, a frequency matrix, a position-specific scoring matrix (PSSM), or a position weight matrix (PWM) [1, 2]. Based on the motif representation used, the algorithms to predict TFBSs fall into two categories: those based on a consensus sequence representation and those based on a matrix representation. Since a matrix representation is considerably more general and more descriptive than its consensus sequence counterpart, we only consider algorithms based on matrix representations in this paper. Examples of such algorithms include AlignACE [3], MEME [4], Bioprospector [5], MDscan [6], YMF [7], Weeder [8], and

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 329–337, 2007.

many more. Different searching strategies were applied is these algorithms. Each algorithm has a motif scoring function that it attempt to optimize. Several popular scoring functions include IC (information content) [9], MAP (maximum *a posteriori* probability) score [6], Group Specificity Score [3], LLBG (least likely under the background model) [10], and Bayesian scoring function [11]. The basic idea behind all these algorithms is to look for sequence signatures in the upstream region of a given gene set that are statistically overrepresented as compared to a reference set of genes or to the genome background.

Although all the algorithms described above have their successful applications, they are far from perfect [12-14]. The common problem is that each algorithm reports a lot of motifs and it is very hard to verify all of them. It is very likely that a large fraction of these predictions are false positives. The first goal is to reduce the false positives in the predictions, and the second goal is to rank the motifs in a biologically meaningful way.

Comparative genomics data is a relatively new source of data which can help to improve the motif prediction. It is well known that motifs that are conserved in orthologous sequences are more likely to be functional. So motif conservation across related genomes can be used to measure biological significance. The degree of conservation of the binding sites among the evolutionarily-related multiple genomes is a crude surrogate for the significance of a motif. It is clear that the increasing number of whole genome sequences of evolutionarily-related organisms can provide additional support to the predictions of binding sites [15-18]. The "phylogenetic footprinting" strategy has be used to find motifs that are conserved across related organisms [19]. However, any motif that is unique for a particular species will not be detected by this method. Furthermore, not all genes have orthologs in all related organisms. Another approach is to treat the orthologous sequences in the same way as sequences that are not phylogenetically related, that is, simply pool the co-regulated genes and their orthologous counterpart before applying computational methods. Obviously, the orthology information is under-utilized in such an approach. Several subtle approaches such as EMnEM [20] (EM-based), PhyloCon [21] (Greedy algorithm), PhyME [22] (EM-based), PhyloGibbs [23] (based on Gibbs Sampling) were developed recently to resolve this problem while taking the phylogenetic relationships into account. However, phylogenetic relationships are notoriously difficult to infer and are often unreliable. Furthermore, most of them need the input sequences to be preprocessed to obtain alignments, which again are potentially unreliable. Inaccurate alignments (or phylogenies) hurt the ability to make accurate predictions.

Unlike the *de novo* motif detection approaches, the method described in this paper refines any given motif by incorporating comparative genomics data. It first searches sequences for candidate sites. And then filters out those that are not the instances of the motif. Recently, Comin *et al.* reported a subtle motif discovery method using a similar two-step strategy [24]. The differences are twofold. First, we incorporate comparative genomics data, and second, we use profiles instead of consensus sequences to represent the motifs.

It is important to note that our method can refine a motif starting from one or more known (perhaps imprecisely specified) binding sites. This makes sense in many cases where biological experiments may have reported partial information wet lab experiments from laboratories are often very focused and it is common that only few genes and even fewer binding sites are identified by such studies. A good starting point for most motif detection methods is the results of gene expression studies, which can help to identify co-expressed (and, therefore, potentially co-regulated) sets of genes. Then using the method described in this paper, and starting from one or two experimentally verified binding sites, we can predict the rest of the relevant binding sites of the genes in the pathway and also output a refined profile for the motif.

2 Methods

2.1 Enhanced Motif Refinement (EMR Algorithm)

One simple approach to integrate comparative genomics data into existing motif-finding methods is to filter out motifs not conserved across related organisms. However, this approach is too simplistic and only considers comparative genomic data in a limited way. Our proposed method considers comparative genomics data in a dual manner. Comparative genomics data is used not merely to filter out motifs, but also to filter out individual sites that are not instances of the motif. Success of *ab initio* methods is often dictated by the signal-to-noise ratio of the input sequences, which is often rather low. Thus, filtering out the individual sites that are not instances of the motif is necessary and meaningful because these sites determine the profile of the motif.

Our algorithm is described below in Figure 1. The algorithm takes as input any motif discovered in a given genome Γ_1 using any motif detection method. Using one or more additional genomes Γ_2, the algorithm returns an enhanced motif.

The motif scoring function optimized in the EMR algorithm is the MAP (maximum *a posteriori* probability) score.

```
Input: a) Motif PWM $M_1$, motif length $l$, and associated gene set $G_1$ from genome $\Gamma_1$,
    b) upstream sequences of the ORFs in $G_1$,
    c) Additional genome(s) $\Gamma_2$,.and the orthology map for all the genomes, and
    d) upstream sequences of the ORFs in $G_2$, the orthologs of $G_1$ in $\Gamma_2$.
Output: Refined motif PWM $M_r$
Algorithm:
1.  Scan the upstream sequences of gene set $G_2$ in genome $\Gamma_2$ with motif $M_1$ for
    instances.
2.  Add all instances found in $\Gamma_2$ to $M_1$ and update motif, if doing so increases motif
    score.
3.  Scan the upstream sequences of gene set $G_1$ with the updated motif.
4.  Again, add instances of motif found in $G_1$ and update motif, if doing so increases
              motif score.
5.  Remove instances of motif and update motif, if doing so increases motif score.
6.  Return updated motif.
```

Fig. 1. EMR Algorithm

The EMR algorithm uses two main subroutines. One uses the motif described as a PWM and finds all instances of it in a given set of sequences. This is achieved by sliding a window of length l and scoring the windows with the PWM, and checking if the score is above an appropriate threshold. Given a sequence S, denoted by $s_1s_2...s_l$, the following quantity given by

$$W_s = \ln \frac{P(S \mid M)}{P(S \mid B)}$$

measures the ratio of P(S|M), the probability of generating S under model M, to P(S|B), the corresponding probability under a background model. Thus if w_{ij} is the entry of PWM corresponding to base r at position i, then

$$W_s = \sum_{i=1}^{l} (w_{ij}).$$

We denote by w_{ir} the entry of base r at position i in the PWM, where base r is the ith base of the motif consensus sequence c. And we define

$$W_c = \sum_{i=1}^{l} w_{ir}.$$

Obviously, W_c is the weight of consensus sequence of the motif, and is the largest weight among all segments. Now we define a ratio R_{sc} between W_s and W_c as similarity of segment S to motif M.

$$R_{sc} = \frac{W_s}{W_c}$$

Another implementation related issue is how to choose a motif scoring function. Many scoring functions have been used in previous motif discovery methods. Motif discovery methods attempt to optimize an appropriate scoring function. A widely used scoring function is the *MAP* (maximum *a posteriori* probability) score, which is a combination of the negative entropy of the PWM and the rareness of the PWM with respect to the background [6].

$$MAP = \frac{\log(x_m)}{l} (\sum_{i=1}^{l} \sum_{j=1}^{4} w_{ij} \log w_{ij} - \frac{1}{x_m} \sum_{alls} \log(P_0(s))),$$

where l is the width of the motif, x_m is the number of sites used to construct the motif, W_{ij} is the entry of base j at position i in the PWM and $P_0(s)$ is the probability of generating the candidate site s from the background model. *MAP* score maintains a good balance well between motif specificity and motif diversity and seems to improve the performance of motif finding algorithm [6]. Therefore it was used in this paper.

2.2 Motif Ranking

The advantage of EMR algorithm is that it refines the motif predictions resulting from any of the current algorithms. Previous research has shown that different motif discovery methods often complement each other [25]. That is, many motif discovery algorithms often find motifs that are missed by other algorithms. Motif finding algorithms search for motifs that maximize different metrics. As a result, motifs judged significant by one algorithm may not be ranked as highly by the others. Scoring functions like the *MAP score* are good for optimization methods, but are not necessarily good for ranking motifs.

In this paper, we consider metrics (and develop new ones) that are better suited for methods that use comparative genomics data. The hypothesis of these methods is that motifs which are conserved in orthologous sequences are more likely to be functional, i.e., biologically significant. The straightforward conservation score was defined as the average number of orthologous upstream sequences in which the motif is found across the genome. Such conservation score was used by Getz *et al.* [17]. In that paper, each gene that has motif sites receives a score that is equal to the number of species in which the corresponding ortholog contained a site in its upstream sequence. The scores for all the genes that contained at least one site in reference genome are averaged together to give the conservation score for a particular motif. In brief, the conservation score S is defined as:

$$S = \sum_{i=1}^{n} \frac{s_i}{n},$$

where s_i is the number of species in which the ortholog of gene i contained a site in its upstream sequence, and n is the total number of genes in the reference genome whose upstream sequence has at least one site of the motif. The weakness of this conservation score is that it does not consider the following key facts:

Fact 1: if m denotes the number of species in comparative genomic data, the more instances in which n_i equals to m, the more significant the motif is;

Fact 2: with the same conservation score, the motif with larger value of n is more significant.

After taking the above two facts into account, we propose a new metric to measure the conservation score and use it to rank the motifs resulting from the EMR algorithm. The new metric is given by:

$$S_c = \log\left[mn\right]\frac{\sum_{i=1}^{m} w_i i n_i}{n\sum_{i=1}^{m} w_i}, w_i > w_{i-1}, \forall i$$

where m is the number of species in comparative genomic data, n is the total number of genes in the reference genome whose upstream sequence has at least one site of the motif, n_i is the number of genes that has i number of species in which the corresponding ortholog contained at least one motif site in the upstream sequence, and w_i is the weight constant which satisfies $w_i > w_{i-1}$ for all i, i. e., higher significance is assigned to motif instances that occurs in the orthologs of more genomes.

3 Results and Analysis

3.1 Synthetic Data

3.1.1 Constructing Synthetic Data

The input to the EMR algorithm are the upstream sequences of ORFs of the reference genome and the upstream sequences of ORFs from genomes of phylogenetically related organisms. The yeast genome was used to constructe a synthetic data set to evaluate the EMR algorithm, we used the data set for the metabolic phase in sporulation, which was complied by Narasimhan et al., from the work of Chu *et al.* [26, 27].

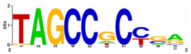
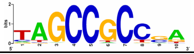
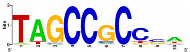
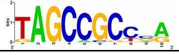
Diploid cells of yeast produce haploid cells through the developmental program of sporulation, which is characterized by sequential transcription of at least four sets of genes — early, middle, mid-late, and late [26, 28]. A conserved site (URS1) was found in the upstream region of many of the known early genes [29]. A total of 42 genes were compiled from the work of Chu *et al.* following a pattern of expression characterized by early induction. The upstream regions of 15 of these genes contained a URS1 motif. The upstream sequences of these 42 genes were obtained as the first step in constructing the synthetic data. All orthologous sequences corresponding to the 42 sequences were extracted from the genomes - *S. paradoxus*, *S. mikatae*, and *S. bayanus* [30]. The gene set associated with these orthologous sequences serve as the gene set G_2 in the EMR algorithm. For keeping the phylogenetic information purpose, the upstream sequence of gene set G_2 remained unchanged. The upstream sequences of gene set G_1 were constructed from the 15 binding sites. A total of 65 random sequences of length 800bp were generated based on the GC content of yeast genome *Saccharomyces cerevisiae*. One copy of a known 15 URS1 binding sites was inserted in a random location of each of the 15 sequences.

3.1.2 Results from Synthetic Data Using EMR Algorithm

The generated sequences (described above) were fed into the motif finding programs AlignACE and MEME. The predictions of motif URS1 from AlignACE and MEME were then refined using the EMR algorithm. The results are shown in Table 1.

From the results summary from Table 1, we can see that all the characteristics of the motif including MAP score, information content, and number of false sites, showed improvement after refinement by the EMR method, regardless of whether the input was the results of using AlignACE or MEME. Although the EMR algorithm attempts to optimize the MAP score, the IC (information content) score of the motif also showed improvement. The data set fed to motif finding algorithms had low signal-to-noise ratio (0.23 in this example), which make it difficult for motif finding algorithm to discover real motif instances. For example, the output of AlignACE found 6 false sites, while

Table 1. Refined motif from synthetic data (the characteristics of motif before and after refinement using the EMR algorithm. The characteristics include consensus sequences or motif logos, MAP scores, information content, and number of false sites).

Program	Known consensus, predicted motif logo, and refined motif logo	MAP Score (Before/After)	Information Content (Before/After)	No. of false sites (Before/After)
AlignACE	TAGCCGCCGA  	2.02/3.63	6.85/8.49	6/2
MEME	TAGCCGCCGA  	2.81/3.64	6.70/8.02	2/2

MEME introduced 2. After refinement, the number of false sites reduced to 2 using the output of AlignACE, and remained unchanged using the output of MEME.

3.2 Results of Yeast Data

In a second set of experiments, we applied the EMR algorithm to real yeast data. The motif data were obtained from the work of Narasimhan et al. [27]. Four known binding sites were investigated including are EBF1, GAL4, PDR, and URS1. The data set consisted of upstream sequences of a set of genes. The details of data sets are summarized in Table 2.

Table 2. Yeast Motif Data Sets Summary

Motif	Total No. of sequences	No. of sequences that have motif hits	Known consensus
EBF1	16	16	ACACCCA
GAL4	10	10	CGGNNNNNNNNNNNCCG
PDR	11	11	TCCGCGGA
URS1	20	47	TAGCCGCCGA

Table 3. Motif Ranking by conservation score Sc

Motif	Rank reported by MEME	Rank reported by EMR	S_c Score*
EBF1	3	1	1.0276 (0.8187, 0.7756, 0.7322, 0.7971)
GAL4	1	1	0.9672 (0.7284, 0.3548, 0.4359, 0.3054)
PDR	1	1	0.6176 (0.2543, 0.1324, 0.2180, 0.4560)
URS1	3	1	0.9168 (0.8198, 0.8686, 0.8321, 0.5123)

* The score outside the parenthesis is the conservation score for the motif listed in the corresponding first column. The score within parenthesis is the conservation score for four other motifs reported by MEME.

Motif finding program MEME was applied to these four data sets. For each data set, five highest-rank motifs based on the metric used by MEME were chosen to report. The motifs listed in Table 3 were expected to be reported as the highest one in each corresponding category. But this is not always the case as shown in Table 3. However, after refinement by our method, the motifs listed in Table 3 were output as the top list. Table 3 summarizes the results.

4 Discussion

Comparative genomics data can play an important role in motif finding and motif ranking. Our work shows that comparative genomics data provides significant improvements in these two areas. Our proposed EMR algorithm helps improve motif prediction by removing artificial upstream segment.

Many variants of motif-finding algorithms have been proposed by researchers. Different algorithms optimize different scoring functions. Our proposed new metric measures the motif score by integrating the comparative genomics data. Previous research shows that evolutionarily conserved motifs are functionally relevant. Our results show that by using comparative genomics data in a sophisticated manner, motif conservation score calculated by our method gives consistent motif ranking and teases out biologically-relevant motifs. When motif finding algorithms are applied to a whole genome sequence, it generated a lot of output with a large number of false positives. Our proposed metric helps to rank these genome-wide motifs to facilitate researchers to choose the best ones for experimental verification.

Acknowledgements. The work of G. N. was supported in part by a grant from NIH under NIH/NIGMS S06 GM008205. E. Z. was supported by Florida International University Presidential Fellowship.

References

1. Stormo, G.D., DNA binding sites: representation and discovery. Bioinformatics, 2000. 16(1): p. 16-23.
2. Werner, T., Models for prediction and recognition of eukaryotic promoters. Mamm Genome, 1999. 10(2): p. 168-75.
3. Hughes, J.D., et al., Computational identification of cis-regulatory elements associated with groups of functionally related genes in Saccharomyces cerevisiae. J Mol Biol, 2000. 296(5): p. 1205-14.
4. Bailey, T.L. and C. Elkan, Fitting a mixture model by expectation maximization to discover motifs in biopolymers. Proc Int Conf Intell Syst Mol Biol, 1994. 2: p. 28-36.
5. Liu, X., D.L. Brutlag, and J.S. Liu, BioProspector: discovering conserved DNA motifs in upstream regulatory regions of co-expressed genes. Pac Symp Biocomput, 2001: p. 127-38.
6. Liu, X.S., D.L. Brutlag, and J.S. Liu, An algorithm for finding protein-DNA binding sites with applications to chromatin-immunoprecipitation microarray experiments. Nat Biotechnol, 2002. 20(8): p. 835-9.
7. Sinha, S. and M. Tompa, YMF: A program for discovery of novel transcription factor binding sites by statistical overrepresentation. Nucleic Acids Res, 2003. 31(13): p. 3586-8.

8. Pavesi, G., et al., Weeder Web: discovery of transcription factor binding sites in a set of sequences from co-regulated genes. Nucleic Acids Res, 2004. 32(Web Server issue): p. W199-203.
9. Hertz, G.Z. and G.D. Stormo, Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. Bioinformatics, 1999. 15(7-8): p. 563-77.
10. Friberg, M., P. von Rohr, and G. Gonnet, Scoring functions for transcription factor binding site prediction. BMC Bioinformatics, 2005. 6: p. 84.
11. Jensen, S.T. and J.S. Liu, BioOptimizer: a Bayesian scoring function approach to motif discovery. Bioinformatics, 2004. 20(10): p. 1557-64.
12. MacIsaac, K.D. and E. Fraenkel, Practical strategies for discovering regulatory DNA sequence motifs. PLoS Comput Biol, 2006. 2(4): p. e36.
13. Sandve, G.K. and F. Drablos, A survey of motif discovery methods in an integrated framework. Biol Direct, 2006. 1: p. 11.
14. GuhaThakurta, D., Computational identification of transcriptional regulatory elements in DNA sequence. Nucleic Acids Res, 2006. 34(12): p. 3585-98.
15. van Nimwegen, E., et al., Probabilistic clustering of sequences: inferring new bacterial regulons by comparative genomics. Proc Natl Acad Sci U S A, 2002. 99(11): p. 7323-8.
16. Xie, X., et al., Systematic discovery of regulatory motifs in human promoters and 3' UTRs by comparison of several mammals. Nature, 2005. 434(7031): p. 338-45.
17. Gertz, J., et al., Discovery, validation, and genetic dissection of transcription factor binding sites by comparative and functional genomics. Genome Res, 2005. 15(8): p. 1145-52.
18. Gertz, J., J.C. Fay, and B.A. Cohen, Phylogeny based discovery of regulatory elements. BMC Bioinformatics, 2006. 7: p. 266.
19. Blanchette, M. and M. Tompa, Discovery of regulatory elements by a computational method for phylogenetic footprinting. Genome Res, 2002. 12(5): p. 739-48.
20. Moses, A.M., D.Y. Chiang, and M.B. Eisen, Phylogenetic motif detection by expectation-maximization on evolutionary mixtures. Pac Symp Biocomput, 2004: p. 324-35.
21. Wang, T. and G.D. Stormo, Combining phylogenetic data with co-regulated genes to identify regulatory motifs. Bioinformatics, 2003. 19(18): p. 2369-80.
22. Sinha, S., M. Blanchette, and M. Tompa, PhyME: a probabilistic algorithm for finding motifs in sets of orthologous sequences. BMC Bioinformatics, 2004. 5: p. 170.
23. Siddharthan, R., E.D. Siggia, and E. van Nimwegen, PhyloGibbs: a Gibbs sampling motif finder that incorporates phylogeny. PLoS Comput Biol, 2005. 1(7): p. e67.
24. Comin, M. and L. Parida. Subtle Motif Discovery for Detection of DNA regulatory sites. in Asia Pacific Bioinformatics Conference (APBC2007). 2007. Hong Kong.
25. Tompa, M., et al., Assessing computational tools for the discovery of transcription factor binding sites. Nat Biotechnol, 2005. 23(1): p. 137-44.
26. Chu, S., et al., The transcriptional program of sporulation in budding yeast. Science, 1998. 282(5389): p. 699-705.
27. Narasimhan, C., P. LoCascio, and E. Uberbacher, Background rareness-based iterative multiple sequence alignment algorithm for regulatory element detection. Bioinformatics, 2003. 19(15): p. 1952-63.
28. Mitchell, A.P., Control of meiotic gene expression in Saccharomyces cerevisiae. Microbiol Rev, 1994. 58(1): p. 56-70.
29. Rubin-Bejerano, I., et al., Induction of meiosis in Saccharomyces cerevisiae depends on conversion of the transcriptional represssor Ume6 to a positive regulator by its regulated association with the transcriptional activator Ime1. Mol Cell Biol, 1996. 16(5): p. 2518-26.
30. Kellis, M., et al., Sequencing and comparison of yeast species to identify genes and regulatory elements. Nature, 2003. 423(6937): p. 241-54.

Mining Discriminative Distance Context of Transcription Factor Binding Sites on ChIP Enriched Regions

Hyunmin Kim[1], Katherina J. Kechris[2], and Lawrence Hunter[1]

[1] Center for Computational Pharmacology, University of Colorado Health Sciences Center,
Aurora, CO 80045 USA
[2] Department Preventive Medicine and Biometrics, University of Colorado Health Sciences
Center, 4200 East Ninth Avenue, B-119
Denver, CO 80262 USA
{Hyun.Kim,Katerina.Kechris,Larry.Hunter}@UCHSC.EDU

Abstract. Genome-wide identification of transcription factor binding sites (TFBSs) is critical for understanding transcriptional regulation of the gene expression network. ChIP-chip experiments accelerate the procedure of mapping target TFBSs for diverse cellular conditions. We address the problem of discriminating potential TFBSs in ChIP-enriched regions from those of non ChIP-enriched regions using ensemble rule algorithms and a variety of predictive variables, including those based on sequence and chromosomal context. In addition, we developed an input variable based on a scoring scheme that reflects the distance context of surrounding putative TFBSs. Focusing on hepatocyte regulators, this novel feature improved the performance of identifying potential TFBSs, and the measured importance of the predictive variables was consistent with biological meanings. In summary, we found that distance-based features are better discriminators of ChIP-enriched TFBS over other features based on sequence or chromosomal context.

Keywords: transcription factor, ChIP-chip data analysis, ensemble learning.

1 Introduction

Transcription factors (TFs) play a key role in transcription, an initial step of gene expression regulation. By binding to *cis*-regulatory DNA sequences (transcription factor binding sites, or TFBSs), these proteins directly control the level of gene expression in a cell. In higher eukaryotes, complexity is introduced with combinatorial interactions of transcription factors, as well as remodeling chromatin structure, and the degree of specific regulation of gene expression varies according to cell type, developmental stage, disease state and environment. A cluster of interacting functional elements of TFBSs is typically called a *cis*-regulatory module (CRM) [1]. Recently, high-throughput experimental techniques such as microarrays and Chromatin ImmunoPrecipitation, followed by detection on genomic microarray chips (ChIP-chip) are available to localize the TFBSs associated with the regulation of expression under

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 338–349, 2007.

specific conditions. Localizing TFBSs is a primary step for the global characterization of gene regulatory networks of multiple genes. Towards this goal, the ENCODE (Encyclopedia of DNA elements) project started in 2004 to identify the functional elements in 1% of the human genome (The ENCODE Project Consortium 2004).

Because of protein-protein interactions between transcription factors, their position and orientation are functionally tied within a CRM. To characterize CRMs, putative TFBSs (pTFBSs) need to be calculated. One of the current computational strategies is to discover motifs conserved in a particular set of CRMs [2, 3]. Another approach is to search for the known binding sites using a library of previously characterized profiles called position weight matrices (PWMs) [4, 5]. Given a DNA sequence, a pTFBS is defined as a hit of the corresponding PWM when the PWM score satisfies the predetermined threshold. Many machine learning techniques and statistical models have been developed to characterize the pTFBSs in the CRM context. Regression methods incorporate motif occurrences of tissue-specific factors with cooperation rules and spacing constraints between pTFBSs (for summary, see [6]). The other aspect of CRM modeling deals with characterizing the distance effects of interacting TFBSs [7, 8]. Current efforts include the discrimination of pTFBS patterns in ChIP enriched regions [6, 9, 10], in which the DNA sequence region of a high ChIP enrichment score is defined as a ChIP-oriented CRM (cCRM).

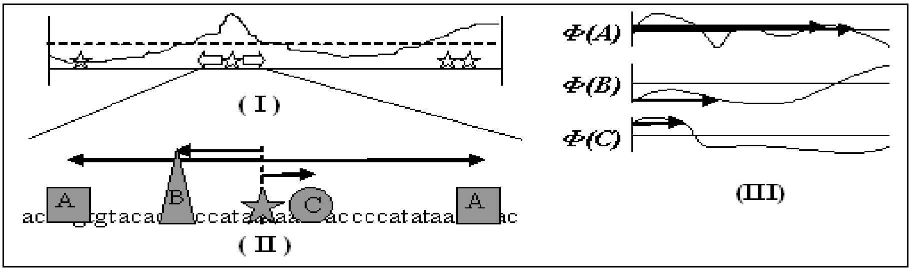


Fig. 1. Overview of characterizing distance features. (I) Shown are pTFBSs of a target TF (marked with stars), and the ChIP enrichment scores (solid line) of the corresponding TF with the threshold (dotted line). (II) The target cCRM expands in each direction from the center of the target pTFBS (star). Thick arrows depict the distance of a neighboring TFBS from the center of the target pTFBS. (III) For each TFBS, positioning in a particular distance is illustrated with a thick arrow, and a score of the positioning is calculated by the corresponding odds ratio function of a distance value (see materials and methods section).

Unlike conventional approaches, we examine the context of a cCRM represented by distance preferences of surrounding pTFBSs calculated by PWMs (Fig. 1). In this study, we selected the ChIP-chip experimental data sets produced by the ENCODE project for identifying target TFBSs of HNF4α, HNF3β, and USF1 in the HepG2 (liver) nuclear extracts [11]. Using ensemble rule learners, we characterize this context, and compare it with other types of contexts including pTFBS counts, and genomic contents (e.g., DNase Hypersensitive sites and nucleosome binding occupancy), associated with a conformation of chromatin structure. The evaluation procedure is performed using five-fold cross validation on the average error rates. Based on the best-performed learner, we further analyze the importance and interaction effects of variables used in each context.

2 Materials and Methods

We introduce a variety variables types to represent the context of a cCRM. The first variable type reflects the number of occurrences of a pTFBS (count variable). The second type represents a distance preference between pTFBSs (distance variable). The last type involves genome contents such as DNase hypersensitive sites, multi-species conservation, nucleosome occupancy, and histone acetylation sites (genome content variable). Using different combinations of the above variable types, we build classifiers for testing the relative contribution of a particular variable set.

2.1 Prediction of Putative TFBSs in the ENCODE Genomic Region

The forty-four ENCODE (ENCyclopedia Of DNA Elements) regions comprise 1% (30 Mb) of the human genome. Non-repeat ENCODE regions in hg17 version of UCSC genome browser were downloaded [12]. The prediction of TFBSs (putative TFBSs, or pTFBSs) were performed using the MATCH program with TRANSFACv9.4 vertebrate PWM profiles [4]. After masking the genomic DNA sequences in the repeat and exon regions, the binding affinity between the DNA sequence and a TF is measured by PWM scoring [13]. We defined the “hits” of PWM (i.e., putative TFBS or pTFBS) on the DNA sequence when its binding affinity score is equal or greater than the threshold (optimized by the minimum false positive ratio in the MATCH program, see [4] for the details). Mapping information between a TF and a set of the corresponding PWMs was extracted from the transcription factor-to-pwm fields in TRANSFAC.

2.2 Scoring ChIP Enrichment

The ChIP-chip experimental data sets of the target TFs (HNF4α, HNF3β, and USF1) were downloaded from the ENCODE ChIP-chip track in hg17 UCSC genome repository (original paper is in [11]). For each target TF (i.e., an antibody target for a ChIP experiment), a set of DNA sequences (i.e., candidate cCRM) was obtained by expanding positions of the corresponding pTFBSs equally in each direction to have length 1 kb (Fig. 1-I). The resultant regions overlapped by >990 bp were merged to form the final set of candidate cCRMs. We assigned the ChIP score for the candidate cCRMs by taking the ChIP score overlapping with the center position of the cCRMs (Fig. 1-I). Then, following the threshold used in the original paper, the DNA sequences with $\log_2$ ChIP score of greater than or equal to 1.2 were selected as a positive cCRM set. Note, the negative set also contains the target pTFBSs (false positives). The DNA sequences of high ChIP scores without target pTFBSs (false negatives) were ignored for this study.

2.3 Building Distance Variables

Given a positive set S_p and a negative set S_n of cCRMs, we took the distribution of distances between a neighbor pTFBS and the target pTFBS. The distribution was smoothed using Gaussian kernels [14], implemented in the density function (adjust option =0.2) in the statistical package R (v2.4.0). The following log odds ratio form defines the function *Φ* of the distance preference of the *j*th PWM in a distance *d* (see examples in Fig 1-III).

$$\phi(j,d) = \log \frac{\Pr(d \mid PWM_j, S_p)}{\Pr(d \mid PWM_j, S_n)}, \tag{1}$$

where $\Pr(d \mid PWM_j, s)$ is the (smoothed) density of the PWM *j* in the subset *s* of cCRMs with a distance *d*.

The DNA sequence $s \in S$ contains a sequence of tuples $a_1 \ldots a_n$ where each tuple a_k=<a_k^M, a_k^P> denotes the match position *P* of the PWM *M*. Now, this content is transformed into variables **x** as the input of a classifier. Using the above distance preference function *Φ*, we can calculate the variable x_{ij} of PWM *j* in a sequence *i*:

$$x_{ij} = \frac{\sum_{k=1}^{n} \phi(a_k^M, |a_k^P - c|) I(a_k^M = j)}{\sum_{k=1}^{n} I(a_k^M = j)}, \tag{2}$$

where *c* is the center position of sequence *i,* and *I* is an indicator function.

2.4 Acquisition of Genome Content Variables

ChIP-chip experiments do not yet have the resolution to recognize precise cis-regulatory elements (DNA sequences) that may be bound by TFs. Computational methods designed for searching known elements suffer from extremely high false positive rate [15, 16]. Additional genome contents would be useful to filter out such false positives in the cCRMs. These include nucleosome occupancy (NS), multi-species conservation (CN), and DNase hypersensitive sites (HS). The NS variables were calculated using a chicken model [17], and the variables for HS and CN were obtained from the UCSC genome database[1] by taking: raw scores of HS mapping in the HepG2 cell line [18]; union set of consensus element tracks of 28 vertebrate species, and converting to binary values (0 or 1). Phylogenetic foot-printing is well suited for identifying evolutionarily conserved sequences [19, 20], however it cannot detect lineage/species-specific TFBSs [21]. The DNase hypersensitive (HS) sites have shown to be well associated with *cis*-regulatory elements including promoters, enhancers, suppressors, insulators, and locus control regions. This approach has the advantage of considering chromatin context, allowing for identification of both ubiquitous and tissue-specific regulatory elements.

2.5 Rule Ensemble Learning

Ensemble learning has emerged as a promising tool in bioinformatics, where weak, noisy signals are common. It has been reported that rule-based ensemble classifiers

[1] http://hgdownload.cse.ucsc.edu/goldenPath/hg17/encode/database/

perform very well compared to other classifiers such as discriminant analysis, support vector machines, and neural networks [22]. The ensemble classification model is constructed as linear combinations of simple rules. Each rule consists of conjunction of decision statements considering the value of individual input variables. The decision statements are easy to understand and used to assess the degree of relevance of the respective input variables. The unknown value of an attribute *y,* denoting ChIP enrichment, is predicted using known joint values of *n* variables (of different types) $x = (x_1, x_2, \ldots, x_n)$. The predictive model $\hat{y} = F(x)$ of the ensemble learner [23] is denoted by

$$F(x) = \hat{a}_0 + \sum_{k=1}^{K} \hat{a}_k r_k(x) + \sum_{j=1}^{n} \hat{b}_j l_j(x_j) \tag{3}$$

where $r_k(x)$ and $l_j(x)$ are basis functions of *K* conjunctive rules and *n* original variables respectively. Given a loss function *L(y,F)*, the parameters are estimated with

$$(\{a_k\}_0^K, \{b_j\}_1^n) = \underset{\{a_k\}_0^K, \{b_j\}_1^n}{\arg\min} \sum_{i=1}^{N} L(y_i, F(x_i)) + \lambda(\sum_{k=1}^{K} | a_k | + \sum_{j=1}^{n} | b_j |) \tag{4}$$

where the first term in (4) measures the prediction risk, and the second term penalizes large values for the coefficients of the base learners. Both RandomForests and RuleFit algorithms are ensemble learners and variants of the above model with different usages of basis functions and definitions of the loss function. We obtained the RandomForests program from the randomForests library in the R 2.4.0 package, and the beta version of RuleFit from the website[2].

2.6 Tuning Classifiers for the Imbalanced Data

Our data set is extremely imbalanced (e.g., 200 positive examples versus 8000 negative examples in the HNF4α data), which may result in a classifier with poor performance for predicting the minority class. There are two common solutions to handle the problem. One is based on cost sensitive learning by assigning a high cost to a misclassification of the minority class to minimize the overall costs. The other one is a sampling approach forcing equal class priors by down-sampling the majority class or over-sampling the minority class (see applications using different classifiers [24-26]). To handle the imbalance between positive and negative sets, we employed the down-sampling scheme for RandomForests and cost sensitive (assigning a high cost to misclassification of minority class) method for the RuleFit. We defined the balancing factor (BF) as a rate proportional to the size of the minority class. In RandomForests, down samples of #minority samples for minority class and #minority samples/BF for majority class were applied. In RuleFit, class weight of BF/#minority samples for minority samples and 1/#majority samples for majority samples were used. RandomForests has two main parameters mtry and ntree to be optimized. The default setting of ntree=500 performs well and does not increase the out-of-bag (OOB) error. We obtained the optimum parameters for each rebalanced data set, using

[2] http://www-stat.stanford.edu/~jhf/R-RuleFit.html

the tuneRF function in the RandomForests library. We used the default settings for the RuleFit algorithm, and the rebalancing procedure is conducted by applying different BFs.

2.7 Performance Evaluation

RandomForests concurrently estimates the classification error when each tree is constructed using a different bootstrap sample from the original data. About one-third of the samples are left out of the bootstrap sample for testing the error rate of the tree. The aggregated error rates are averaged, and called the out-of-bag (OOB) estimate of error rate. In general cases, we separate the training and testing sets to estimate the classification error using the k-fold cross validation method. We fit the classifiers until they achieve the balanced error rate (equal or similar error rates on both positive and negative sets), and then take the average error rates to report and compare the performance. The positive error rate corresponds to the false positive error rate (FP/T), or 1-sensitivity, and the negative error rate corresponds to the false negative error rate (FN/N), or 1-specificity.

2.8 Variance Importance and Interaction Criteria

The RandomForests algorithm estimates the importance of a variable by measuring the increasing prediction error when that variable is permuted while all others are left unchanged. We used the “Gini importance” criterion available in the RandomForests implementation [27]. The “Gini importance” describes the improvement in the “Gini gain” splitting criterion. This involves non-linear and complex high-order interaction effects for predictor variables of PWM hits. In the RuleFit algorithm, the relative importance of input variables are judged by their participation in the most influential predictors (rules or linear) appearing in the model [23].

3 Results

The ChIP-chip experimental data sets involve Hnf4α, Hnf3β, and USF1 data sets. ENCODE regions contain 10k to 30k hits of corresponding PWM binding sites

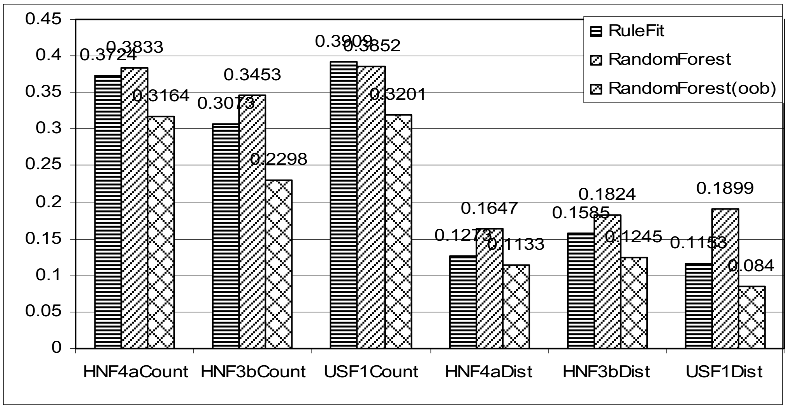
Table 1. Summary of data set statistics. pos: number of positive cCRMs; neg: number of negative cCRMs; miss: number of ChIP enriched sequences without any hits of corresponding PWMs; hits: number of PWM hits matched to the total cCRMs; profile: PWM ids of a target TF.

TF	cCRM				PWM	
target	pos	neg	total	miss	hits	profile
HNF4α	209	8,301	8,510	133	10,004	HNF4_01, COUP_01, HNF4_Q6, DR1_Q3, HNF4_Q6_01, HNF4_01_B, HNF4ALPHA_Q6, HNF4_Q6_02, HNF4_Q6_03, HNF4_DR1_Q3,
HNF3β	272	18,259	18,567	68	28,566	HNF-3b, XFD3_01, HNF3_Q6_01, HNF3_Q6,HNF3B_01
USF1	26	5,701	5,727	17	10,927	USF_C, USF_01, USF_Q6, EBOX_Q6_01, USF_02, USF_Q6_01

(Table 1). The set of sequences is expanded from these hits (see materials and methods section). The configuration of final positive (ChIP enriched) regions is 209, 272, 26 for the Hnf4α, Hnf3β, and USF1 data sets, respectively. Of the positive sequences 33% (113) do not contain PWM hits for Hnf4α (20% for Hnf3β, and 39% for the USF1). These missing DNA sequences could be found by relaxing the threshold for the PWM score or with different PWM profile or motifs. As mentioned in [11], those missing binding sites could be a weak signal in which their overall binding affinity is compensated by direct or indirect binding with other cooperative TFs thru protein-protein interactions. The fraction of DNA sequences are very similar to the original report [11].

3.1 Performance Evaluation for Discriminating ChIP Enriched TFBSs

Over three data sets in Table 1, we evaluated the performance of the selected variable type using the two ensemble classifiers RandomForests and RuleFit. After rebalancing the data sets (see materials and methods section), the performance of the classifiers were estimated by five-fold cross validation (Fig 2).



		count			distance		
algorithm	par	HNF4a	HNF3b	USF1	HNF4a	HNF3b	USF1
RuleFit	BF	6	4	4	10	6	13
RandomForests	BF	1.66	1.65	1.65	1.6	1.55	1.6
	mtry	16	8	4	4	4	2

Fig. 2. Comparison of average error rates using the five-fold cross validation and OOB. RuleFit and RandomForests classifiers using the count variables (left three triplets), and the distance variables (remaining three triplets) in the top panel; The bottom panel summarizes the parameter settings used for the rebalancing; the mtry values are automatically determined by the tuneRF function.

Applying count variables, both predictors misclassified the data sets with about 30% average error rate. The gain is dramatic when we use distance variables as a feature. On average, the error rate of classifiers with distance features ranged from 10-20%. The RuleFit algorithm outperformed the RandomForests in most data sets. This can be explained by the selective nature of the RuleFit model, in which additional linear terms (equation 3), when it properly reflects the characteristics of the class, will improve the performance of the model and owing to the lasso penalty (equation 4) their influence is controlled. In contrast, RandomForests fixes the number of variables (mtry) assigned in a node in building each tree.

3.2 Important TFBS Variables

We examined the importance of 224 predictor variables combining count and distance types thru a RuleFit importance measurement, and selected the top twenty entries in Table 2. The distance variables (notated with the prefix "D.") were ranked higher than most of the count variables (with the prefix "C."), except some PWMs belonging to the target TFs. HNF4α and HNF3β data sets share the important variables with different order. HNF-type genes (e.g., Hnf1β(tcf2), HNF1α(tcf), Hnf4, Hnf3β(Foxa2), Hnf6(Onecut1)), play a role in tissue-specific gene regulation in adult cells, including pancreas, liver, lung, and intestinal gene expression, and these factors appear to cooperate with more tissue-specific regulators C/EBP-α (CCAAT/enhancer binding protein), and Cdx1[28]. These data suggest that the HNF4 factor may be acting by direct communication with distant C/EBP factors. Interestingly, lymphoid enhancer-binding factor 1 (LEF-1) has been reported to have a similar effect regarding bent DNA, facilitating an interaction between proteins bound at the distant 5' and 3' enhancer [29]. C.HNF3_Q6_01 and C.USF_02 indicate an importance of repeated occurrences of the target pTFBSs in HNF3β and USF1data set.

Table 2. Relative importance of feature variables mixture (count + distance). Parenthesized number id represents interacting group (interaction strength > 0.1).

	HNF4α	HNF3β	USF1
1	D.CEBP_Q3 (a,b)	C.HNF3_Q6_01 (a)	C.CMAF_01
2	D.CDPCR1_01	D.HELIOSA_01	C.SREBP1_Q6
3	D.EN1_01 (b)	D.PAX2_02	D.CDXA_01
4	C.CEBPGAMMA_Q6	D.PBX_Q3	C.USF_02
5	D.HNF3_Q6_01	D.AML1_01	D.CDXA_02
6	D.PBX_Q3 (a)	D.TATA_01 (a)	D.SMAD_Q6
7	D.YY1_Q6	D.DBP_Q6	C.COMP1_01
8	D.HNF1_C (c)	D.MAF_Q6_01 (b)	D.PBX1_03 (a)
9	D.NF1_Q6_01	D.SMAD_Q6	C.E2F1_Q3 (a)
10	D.CDX2_Q5	D.TTF1_Q6	D.PAX2_01 (a)
11	D.HMGIY_Q6	D.HELIOSA_02	D.CAP_01
12	D.HNF1_Q6	C.HNF4_01	D.HMGIY_Q3
13	D.TATA_01	D.CEBP_Q2_01	D.AML1_01
14	D.PAX4_02	C.LEF1_Q2 (b)	D.TEF1_Q6
15	D.FOXP3_Q4 (c)	D.CEBP_01(a)	D.OCT1_Q6 (b)
16	D.TEF1_Q6	D.FOXD3_01	D.PAX4_01 (b)
17	D.DR1_Q3	D.CMAF_01	D.PAX4_03 (b)
18	D.PAX2_01	D.CDX2_Q5	
19	D.CDXA_02	D.CDC5_01	
20	D.EFC_Q6	D.EFC_Q6	

3.4 Genome Contents and Related ChIP Experimental Data Set

In this section, the performance of distance variables is compared with those of variables involving genome contents (GCs) and the related ChIP-chip experiments (ChIP) other than the target ChIP-chip experiment. Fig 3 represents performances of each individual variable type (left panel), and the relative importance of a variable in each experiment (right panel). Our distance feature performed best overall. However, none of distance variables is more influential than the best ChIP/genome content feature (e.g., hnf3b in HNF4α data set, hnf4a for HNF3β data set, ach3(acetylation genome content) for USF1 data set in Fig. 3-right). It is not surprising because, for example, the Hnf4α and Hnf3β ChIP-chip data sets are strongly correlated (Pearson correlation = 0.670, pval < 2.2e-16).

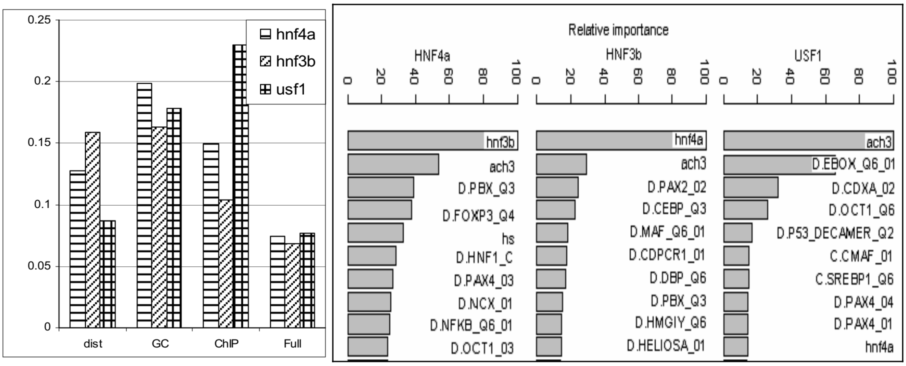


Fig. 3. Performance of RuleFit model applying various variable types with distance, genome content (GC), and related ChIP score of other target (ChIP). Right panel represents relative importance of variable of the full model.

The full models of all possible variables achieved the performance of 7.4%, 6.8% and 7.7% average error rates, and 98.3, 98.0 and 98.0 for the area under the ROC Curve (AUC) in the HNF4α, HNF3β, and USF1 data sets respectively. Among GC variables (i.e., nucleosome binding sites (ns), DNase hypersensitive sites (hs), multi-species conserved regions (con), and nucleosome acetylation sites (ach3)), ach3 and hs were chosen as the relatively important variables.

We also examined the dependence of the full models on the genome contents variables. Fig. 4 displays partial dependence plots for selected GC variables (see the definition in [23]). For each dataset, the relative importance of GC variables is drawn in the left column, followed by four sub plots displaying the degree of dependence (y axis) for values of different variables (x-axis). The plots for ach3 show a positive linear response for the predictor variables of the HNF4α and HNF3β models and a drastic increase at 0.5 for USF1. Different dependence distributions for hs and ns indicate that there are distinct genome content preferences for different TFs.

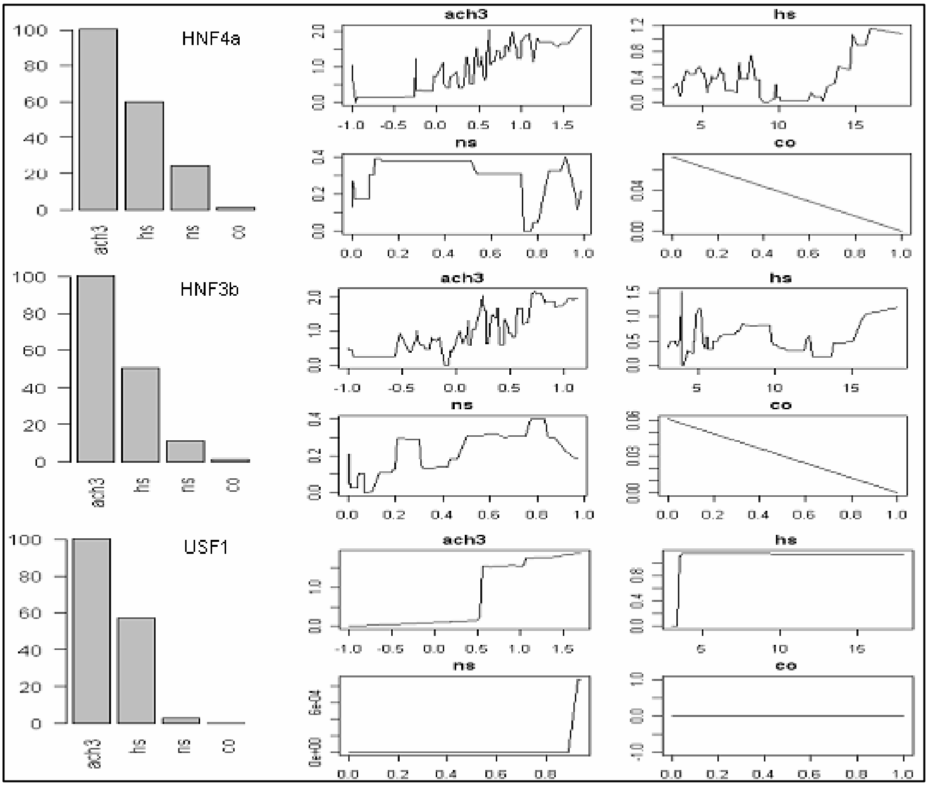


Fig. 4. Relative importance and partial dependence function of genome content variables. Panels in the left column show relative importance of variables. Plots in the right columns display partial dependence plots, which are graphical depictions of the effects of individual variables on the class predictions.

4 Discussions and Conclusions

Regarding the missing ChIP enriched regions (Table 1), ChIP-chip experiments of HNF4α showed that the TFBSs of HNF4α contain small, but distinct motif patterns in promoter regions (within 5k bp of TSS) different from enhancer regions [11]. This may be because some of the HNF4α interactions with proximal promoters might be indirect through formation of enhancer/promoter loops with HNF4α bindings. These TFs are commonly bound to distal regulatory elements and may participate in long distance communication. The pTFBS of HMG-I/Y is chosen as an important variable (Table 2). The HMG-I/Y binds preferentially to the minor groove of A+T rich regions in double stranded DNA. It suggests that HMG-I/Y may function in nucleosome phasing, which in turn changes the topology of chromatin structure. We observed a slight contribution of nucleosome occupancy and multi-species conserved regions. These features are known to be correlated with particular TFs such as Sp1 (data is not shown), and non lineage/species specific TFs, respectively. Therefore, careful consideration is needed to apply the genomic contents as a filtering procedure (for summary of identification of TFBS, see [30]).

Using ensemble learners, we evaluated the usefulness of distance features of pTFBSs, for which effects are composed of a large volume of weak signals, and are consistent with biological meanings. We are testing the method against the remaining 99% of the human genome, and the missing 30% of ChIP enriched regions. This methodology could be extended to the characterization of CRM models for diverse TF roles affecting tissue-specificity or time-dependent expression level of a particular gene set. Combined with other ChIP-chip results, our approach will allow researchers to discover synergistic TF participants for regulation in mammalian systems under different preference of genomic contents preferences.

Acknowledgments

We acknowledge Anis Karimpour-Fard for helpful comments; Sonia Leach for help with an earlier version of algorithm, and David Farrell for help with setting up computer system.

References

1. Yuh, C.H., H. Bolouri, and E.H. Davidson, *Genomic cis-regulatory logic: experimental and computational analysis of a sea urchin gene.* Science, 1998. **279**(5358): p. 1896-902.
2. Bailey, T.L. and M. Gribskov, *Score distributions for simultaneous matching to multiple motifs.* J Comput Biol, 1997. **4**(1): p. 45-59.
3. Roth, F.P., et al., *Finding DNA regulatory motifs within unaligned noncoding sequences clustered by whole-genome mRNA quantitation.* Nat Biotechnol, 1998. **16**(10): p. 939-45.
4. Kel, A.E., et al., *MATCHTM: a tool for searching transcription factor binding sites in DNA sequences.* Nucl. Acids Res., 2003. **31**(13): p. 3576-3579.
5. Sandelin, A., et al., *JASPAR: an open-access database for eukaryotic transcription factor binding profiles.* Nucleic Acids Res, 2004. **32**(Database issue): p. D91-4.
6. Smith, A.D., et al., *Mining ChIP-chip data for transcription factor and cofactor binding sites.* Bioinformatics, 2005. **21**(suppl_1): p. i403-412.
7. Yu, X., et al., *Genome-wide prediction and characterization of interactions between transcription factors in Saccharomyces cerevisiae.* Nucleic Acids Res, 2006. **34**(3): p. 917-27.
8. Yu, X., et al., *Computational analysis of tissue-specific combinatorial gene regulation: predicting interaction between transcription factors in human tissues.* Nucleic Acids Res, 2006. **34**(17): p. 4925-36.
9. Jin, V.X., et al., *A computational genomics approach to identify cis-regulatory modules from chromatin immunoprecipitation microarray data--A case study using E2F1.* Genome Res, 2006. **16**(12): p. 1585-95.
10. Macisaac, K.D., et al., *A hypothesis-based approach for identifying the binding specificity of regulatory proteins from chromatin immunoprecipitation data.* Bioinformatics, 2006. **22**(4): p. 423-9.
11. Rada-Iglesias, A., et al., *Binding sites for metabolic disease related transcription factors inferred at base pair resolution by chromatin immunoprecipitation and genomic microarrays.* Hum. Mol. Genet., 2005. **14**(22): p. 3435-3447.

12. Karolchik, D., et al., *The UCSC Genome Browser Database.* Nucleic Acids Res, 2003. **31**(1): p. 51-4.
13. Wasserman, W.W. and A. Sandelin, *Applied bioinformatics for the identification of regulatory elements.* Nat Rev Genet, 2004. **5**(4): p. 276-87.
14. Silverman, B.W., *Density estimation for statistics and data analysis*. 1986, London: Chapman and Hall.
15. Clifford, S., et al., *Contrasting effects on HIF-1alpha regulation by disease-causing pVHL mutations correlate with patterns of tumourigenesis in von Hippel-Lindau disease.* Hum Mol Genet, 2001. **10**(10): p. 1029-1038.
16. Pennacchio, L.A. and E.M. Rubin, *Genomic strategies to identify mammalian regulatory sequences.* Nat Rev Genet, 2001. **2**(2): p. 100-9.
17. Segal, E., et al., *A genomic code for nucleosome positioning.* Nature, 2006.
18. Crawford, G.E., et al., *Identifying gene regulatory elements by genome-wide recovery of DNase hypersensitive sites.* Proc Natl Acad Sci U S A, 2004. **101**(4): p. 992-7.
19. Thomas, J.W., et al., *Comparative analyses of multi-species sequences from targeted genomic regions.* Nature, 2003. **424**(6950): p. 788-93.
20. Huber, B.R. and M.L. Bulyk, *Meta-analysis discovery of tissue-specific DNA sequence motifs from mammalian gene expression data.* BMC Bioinformatics, 2006. **7**: p. 229.
21. Slightom, J.L., et al., *The complete sequences of the galago and rabbit beta-globin locus control regions: extended sequence and functional conservation outside the cores of DNase hypersensitive sites.* Genomics, 1997. **39**(1): p. 90-4.
22. Breiman, L., *Random forests.* Machine Learning, 2001. **45**(1): p. 5-32.
23. Friedman, J.H. and B.E. Popescu, *Predictive Learning viva Rule Ensembles*. 2005, Department of Statistics, Stanford University.
24. Chen, C., A. Liaw, and L. Breiman, *Using random forest to learn imbalanced data*. 2004, statistics department, university of california at berkeley.
25. Guo, H. and H.L. Viktor, *Learning from imbalanced data sets with boosting and data generation: the DataBoost-IM approach.* SIGKDD Explor. Newsl. , 2004 **6** (1): p. 30-39
26. Akbani, R., S. Kwek, and N. Japkowicz, *Applying support vector machines to imbalanced datasets.* ECML, 2004: p. 39-50.
27. Breiman, L., *Manual on setting up, using, and understanding random forests v3.1*. 2002, http://oz.berkeley.edu/users/breiman.
28. Jensen, J., *Gene regulatory factors in pancreatic development.* Dev Dyn, 2004. **229**(1): p. 176-200.
29. Giese, K., J. Cox, and R. Grosschedl, *The HMG domain of lymphoid enhancer factor 1 bends DNA and facilitates assembly of functional nucleoprotein structures.* Cell, 1992. **69**(1): p. 185-95.
30. Elnitski, L., et al., *Locating mammalian transcription factor binding sites: A survey of computational and experimental techniques.* Genome Res., 2006: p. gr.4140006.

Enhanced Prediction of Cleavage in Bovine Precursor Sequences

Allison N. Tegge[1], Sandra L. Rodriguez-Zas[1,2,3], J.V. Sweedler[3,4],
and Bruce R. Southey[1,4]

[1] Department of Animal Sciences, University of Illinois at Urbana-Champaign, IL 61801, USA
[2] Department of Statistics, University of Illinois at Urbana-Champaign, IL 61801, USA
[3] Institute for Genomic Biology, University of Illinois at Urbana-Champaign, IL 61801, USA
[4] Department of Chemistry, University of Illinois at Urbana-Champaign, IL 61801, USA
{SL,Rodriguez-Zas}rodrgzzs@uiuc.edu

Abstract. Neuropeptides are important signaling molecules that influence a wide variety of biological processes. The prediction of neuropeptides from precursor proteins is difficult due to the numerous and complex series of enzymatic processing and posttranslational modification steps. Bioinformatics prediction of cleavage sites using statistical models was used to overcome the challenge of identifying neuropeptides. Binary logistic models were trained on a bovine dataset and validated on a mammalian dataset that contained no bovine precursors. A model that incorporated amino acid locations and properties provided more accurate and precise cleavage predictions than one using amino acid locations alone. All models consistently resulted in highly accurate predictions of cleavage sites in both datasets. The logistic model proposed can accurately predict cleavage sites in mammalian species and minimize the time consuming and costly experimental validation of neuropeptides.

Keywords: Neuropeptide, precursor, cleavage, logistic regression.

1 Introduction

Efforts to annotate recently sequenced genomes from various organisms have resulted in a comprehensive connection between the genomic and proteomic components for extensively studied species. Genomes of species with less experimental information are being sequenced and in these scenarios, comparative genomics and sequence homology can support a first attempt at annotating the genomes. However, for certain protein and peptide groups such as neuropeptides, similarities between protein sequences from both well and poorly studied species may provide an inaccurate prediction of a bioactive neuropeptide, thereby hindering the functional annotation of predicted genes.

Neuropeptides are small signaling molecules located in the central nervous system and are essential components of the information processing and communication between neurons [1]. They affect a vast number of biological processes including development, growth, reproduction, health, communication, memory, and behavior. Neuropeptides are formed from cleavage of large precursor proteins involving

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 350–360, 2007.

complex posttranslational enzymatic processing. After translation, the precursor is passed into the endoplasmic reticulum and the signal peptide is cleaved by signal peptidases. Further cleavage by multiple peptidases (prohormone convertases or PCs) and other posttranslational modifications (e.g., amidation, glycosylation, phosphorylation) occur before the final bioactive neuropeptides are formed.

Neuropeptide discovery and confirmation is an area of active research; however, experimental verification is time consuming and costly. Consequently, information on precursors and resulting neuropeptides is very limited in many species. The incomplete and highly variable level of neuropeptide characterization across species and neuropeptide families can be traced to different scenarios. A common scenario involves research that targets a specific bioactive neuropeptide with little or no precursor sequence information. In an alternative scenario, the precursor sequence has been identified in multiple species but the resulting neuropeptide(s) are not characterized. Some neuropeptide families have been characterized in many species. For example, insulin (perhaps the best-studied neuropeptide) precursor sequences and cleavage sites are available in 15 mammalian species. In general, approximately half of the neuropeptides found in human, rat and mouse have been experimentally characterized in bovine, the next most-studied mammal.

A first attempt at identifying neuropeptides in species with a limited body of experimental evidence is to use the information from more extensively studied species through sequence homology. However, simple homology can be inadequate to predict neuropeptides because of the complex processing involved in the cleavage of precursors and posttranslational modification(s) of the precursor segments that result in the final neuropeptides. Bioinformatics prediction of precursor cleavage sites using statistical models trained on experimentally confirmed data offers a solution to the challenge of neuropeptide identification. Previously we demonstrated the potential of binary logistic models solely based on the relative location of the amino acids in the precursor sequence to predict the probability of cleavage at any location in the precursor [2], [3], [4], [5], [6] The objectives of this study were to extend the cleavage prediction logistic model by incorporating amino acid properties in addition to amino acids and validate the predictive equation. The enhanced model resulted in improved accuracy and precision in predicting cleavage and provided further insights into the factors influencing cleavage. Because a second draft of the bovine genome is available, the proposed approach was applied to bovine neuropeptides. However, approximately half of the neuropeptide precursors are known in human, rat and mouse have been reported in the bovine. The predictive equations were validated in an independent mammalian dataset of precursors.

2 Materials and Methods

2.1 Data

Mammalian neuropeptide sequences were obtained from the UniProt database (http://www.pir.uniprot.org/). Only mammalian precursors with complete sequence and empirically validated PC cleavage information were considered. The sequences used are available at http://neuroproteomics.scs.uiuc.edu/neuropred.html. The Bovine dataset only included bovine sequences and the Mammalian dataset used to validate the bovine model excluded all bovine sequences. The location of amino acids relative to the cleavage site was denoted following the notation of Schechter and Berger [7]. Amino acids from the scissile bond of any potential cleavage site to the N-terminus were denoted as P, and amino acids towards the C-terminus (opposite side) were denoted as P'. Each amino acid was given a number denoting the location of the amino acid relative to the cleavage site, so that the cleavage occurs between the P1 and P1' amino acids. The physiochemical amino acid properties included in the model for each location were: hydrophobic, hydrophilic, charged, acidic, basic, tiny, small, large, aliphatic, aromatic, sulfur containing, imine, amide, and hydroxyl group containing[8]. The selected precursors were then processed using the following steps described in Southey et al. [5], [6]:

1. Signal peptides were removed using the annotation information or predicted using SignalP V3 [9].
2. Overlapping sliding windows of 18 amino acids from the remaining precursor sequence were created such that every amino acid occurred in the middle or P1 location.
3. Windows without an Arg or Lys in the P1 location and any window with less than four amino acids on either side of the potential cleavage site were removed.
4. Windows that exceeded either the C- or N-terminus of the prohormone were completed with XXX (unspecified amino acid) to ensure the window would be included in the analysis. However, this coding was not used in the training process.
5. When cleavage occurred with multiple sequential basic amino acids, the cleaved site was associated with the most C-terminal basic amino acid.
6. Windows that could not be cleaved due to cleavage at a nearby location were removed. Specifically, windows with a basic amino acid in the P1 location, a basic amino acid present in either the P1' or P4' locations, and no basic amino acid in either the P2 or P4 locations were removed.

2.2 Model

The probability of cleavage, $\pi_{i.}$ within the ith window is described with a binary logistic regression model:

$$logit(\pi_i) = \log(\pi_i / 1 - \pi_i) = \eta_i = \sum_{j=1}^{p} \beta_j \, x_{ij} \; = \mathbf{x}_i' \boldsymbol{\beta}. \tag{1}$$

and

$$\pi_i = exp^{\eta_i} \Big/ 1 + exp^{\eta_i}. \tag{2}$$

where β_j is the regression coefficient associated of the jth model term and x_{ij} denotes the presence or absence of the jth model term in the ith window. A detailed description of logistic regression models can be found in Agresti [10]. Three different sets of model terms were used:

1. individual amino acid at different locations only (Amino Acid model),
2. amino acid properties at different locations only (Properties model), and

3. combined influence of the individual amino acid and the amino acid properties at different locations (Combination model).

Within each set, the initial model terms were identified using a stepwise variable selection approach with minimum significance *P*-value threshold for a term to be included in the model equal to 0.1 and maximum significance *P*-value threshold before a term is removed from the model equal to 0.3. Further model tuning was done to identify the most parsimonious model with the fewest incorrect predictions in the Bovine dataset. The analysis and validation were implemented using the Logistic procedure in SAS (Statistical Analysis Systems, Cary, NC). Artificial neural networks have been used to predict precursor cleavage [11]. Studies comparing the performance of artificial neural networks and logistic models to predict cleavage have demonstrated the superiority of the logistic approach on a wide range of species [3], [4]. Consequently, only results from logistic models will be presented and discussed.

2.3 Validation and Model Accuracy

The final models were validated on precursor sequence information obtained from other non-bovine mammalian precursor sequences in the UniProt database with experimental cleavage data. Mammalian dataset was processed in the same way as the Bovine dataset. Windows were predicted as cleaved if the predicted probability of cleavage exceeded a 50% threshold probability. It was assumed that experimental cleavage information was true and offered a correct depiction of the prohormone processing. This assumption allowed the categorization of predictions into correct cleavage (true-positive result), incorrect cleavage (false-positive result), correct non-cleavage (true-negative result), and incorrect non-cleavage (false-negative result). The following measures of model accuracy [5], [6], [12] were then calculated on all datasets:

1. Correct classification rate: number of correctly predicted sites divided by the total number of sites.
2. Sensitivity (one minus false-positive rate): number of true positives divided by the total number of sites cleaved.
3. Specificity (one minus false-negative rate): number of true negatives divided by the total number of sites not cleaved.
4. Mathews [13] correlation coefficient: a correlation coefficient between observed and predicted cleavage.
5. Positive Precision: number of true-positive results divided by the total number of positives.
6. Negative Precision: number of true-negative results divided by the total number of negatives.

3 Results

There were 48 bovine precursors with an average of 2.58 cleavages per precursor and 288 non-bovine precursors an average of 2.48 cleavages per precursor (66 unique precursors from 26 non-bovine mammalian species) that had complete precursor sequence and sufficient experimental evidence supporting the cleavage sites. After processing, the Bovine dataset consisted of 644 windows including 124 cleavage sites (19% of windows) and the non-bovine Mammalian dataset consisted of 3674 windows including 715 cleaved sites (19% of windows).

Table 1. Number of non-cleaved and cleaved precursor windows by amino acid motif, frequency of the motif and of cleavage in the Bovine and Mammalian datasets

Motif[1]			Bovine Data				Mammalian Data			
P4	P2	P1	NC[2]	C[3]	M Freq[4]	C Freq[5]	NC	C	M Freq	C Freq
K	K	K	1	0	0.002	0.000	19	0	0.005	0.000
K	K	R	2	3	0.008	0.600	5	16	0.006	0.762
K	R	K	1	0	0.002	0.000	5	1	0.002	0.167
K	R	R	2	4	0.009	0.667	16	29	0.012	0.644
K	y	K	18	0	0.028	0.000	110	2	0.030	0.018
K	y	R	14	3	0.026	0.176	81	3	0.023	0.036
R	K	K	0	0	0.000	0.000	2	1	0.001	0.333
R	K	R	2	5	0.011	0.714	11	38	0.013	0.776
R	R	K	2	1	0.005	0.333	12	9	0.006	0.429
R	R	R	6	6	0.019	0.500	35	42	0.021	0.545
R	y	K	24	2	0.040	0.077	142	6	0.040	0.041
R	y	R	29	12	0.064	0.293	151	44	0.053	0.226
y	K	K	15	10	0.039	0.400	103	48	0.041	0.318
y	K	R	10	43	0.082	0.811	61	280	0.093	0.821
y	R	K	17	2	0.030	0.105	116	17	0.036	0.128
y	R	R	30	10	0.062	0.250	159	88	0.067	0.356
y	y	K	157	5	0.252	0.031	841	14	0.233	0.016
y	y	R	190	18	0.323	0.087	1090	77	0.318	0.066
x	K	K	16	10	0.066	0.385	124	49	0.079	0.283
x	K	R	14	51	0.149	0.785	77	334	0.169	0.813
x	R	K	20	3	0.059	0.130	133	27	0.073	0.169
x	R	R	38	20	0.136	0.345	210	159	0.154	0.431
K	x	K	20	0	0.051	0.000	134	3	0.063	0.022
K	x	R	18	10	0.070	0.357	102	48	0.069	0.320
R	x	K	26	3	0.073	0.103	156	16	0.078	0.093
R	x	R	37	23	0.140	0.383	197	124	0.137	0.386
Total			520	124	1.000	0.193	2959	715	1.000	0.195

[1] Motif denotes the amino acid code P4, P2, or P1 location relative to cleavage. L denotes Lysine, R denotes Arginine, x denotes any amino acid and y denotes any non-basic amino acid. The motifs that include x are summations across different motifs.

[2] Number of non-cleaved windows.

[3] Number of cleaved windows.

[4] Proportion of all windows that contain the motif.

[5] Proportion of motif windows that are cleaved.

A summary of the frequency of the amino acid motifs immediately preceding (N terminal) the cleavage site in the training and test datasets is presented in Table 1. The most frequent motifs in the Bovine dataset were xxKR, xxRR and RxxR (where x denotes any amino acid), each observed in 15%, 14% and 14% of windows, respectively (Table 1). However, 78% of the xxKR motifs were cleaved compared to 34% and 38% in the xxRR and RxxR motifs, respectively. These motifs were also the most frequent motifs in the Mammalian dataset although the xxKR and xxRR motifs were associated with a higher proportion of cleavage sites than in the Bovine dataset (81% and 43%, respectively). Approximately 50% of the windows had a basic amino acid in the P1 site and a non-basic amino acid in the P2 or P4 sites in both datasets, with most windows having Arg rather than Lys (32% and 25% of all windows, respectively). In the P1 location, the Bovine dataset had slightly more Lys and Arg windows cleaved (3% and 9%, respectively) than the Mammalian dataset (2% and 7%). Although the xxKK motif was relatively rare, 7% of windows in both datasets, this motif was cleaved 38% and 28% in the Bovine and Mammalian datasets, respectively.

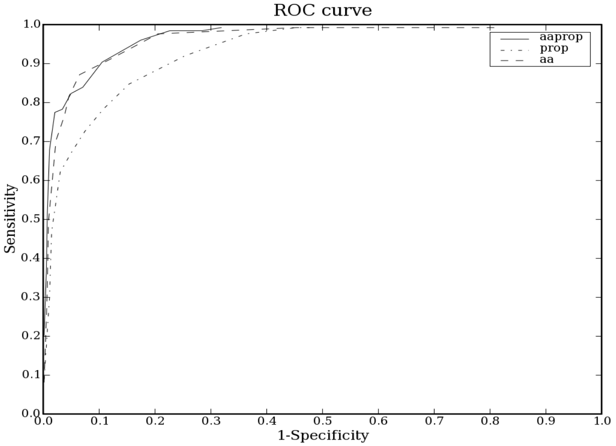


Fig. 1. The ROC curve for the Amino Acid (aa), Properties (prop) and Combined (aaprop) models using the Bovine dataset

All indicators of model accuracy corresponding to the Amino Acid, Properties and Combined models for the Bovine and Mammalian datasets are summarized in Table 2. Each of the three models offered an adequate to very good prediction of cleavage in the Bovine dataset. The Properties model, trained only using amino acid properties, had the weakest performance of all models for all measures except specificity. The Combined model, which included both amino acid properties and specific amino acids, had the best performance of all models except for specificity. The three most significant amino acid positions in the Combined model were: Lys at P2, Arg at P2, and Pro at P1', with the first two having a positive, and the other one a negative, effect on cleavage. Among the most significant amino acid properties and positions in the Combined model, sulfur-containing amino acids at P5 and neutrally charged amino acids in P3', hindered the probability of cleavage, whereas aromatic amino acids at P6' and hydroxyl group containing amino acids at P2' enhanced the probability of cleavage.

Overall, the differences between the best and worst models amounted to eight false-positive and 14 false-negative cleavage predictions. These results indicated that at least 90% of windows would be correctly classified by any of the models. Furthermore, at least 92% of windows would be correctly predicted to be non-cleaved and at least 76% of windows would be correctly predicted to be cleaved. In Figure 1, the ROC curve of the three models tested on the Bovine dataset is shown. This graph shows that high levels of sensitivity can be obtained before a drop in specificity is observed, confirming the accuracy of the models.

The same relative performance among models observed in the bovine training was also observed in the Mammalian dataset (Table 2). In general, the Combined model had the best model accuracy and the range in performance indicators among models was higher in the Mammalian dataset than in the Bovine dataset. For example, the difference between the sensitivity of the Properties and Combined models was 0.16 in the Mammalian dataset.

Table 2. Indicators of model performance for the Amino Acid (AA), Properties (Prop) and Combination (Comb) models in the training (Bovine) and test (Mammalian) datasets

Measure	Bovine Data			Mammalian Data		
	AA	Prop	Comb	AA	Prop	Comb
TN[2]	95	83	97	514	402	493
FN[2]	500	494	502	2698	2699	2768
TP[2]	20	26	18	261	260	191
FP[2]	29	41	27	201	313	222
CCR (%)[3]	92.4	89.6	93.0	87.4	84.4	88.8
Corr (%)[4]	74.9	65.1	77.0	61.2	48.9	63.6
Sen (%)[5]	76.6	66.9	78.2	71.9	56.2	69.0
Spec (%)[6]	84.0	85.6	83.8	84.0	87.0	84.9
PPrec (%)[7]	82.6	76.1	84.3	66.3	60.7	72.1
NPrec (%)[8]	94.5	92.3	94.9	93.1	89.6	92.6

[1] Cleavage was predicted using a 50% threshold probability.

[2] Number of True Negative (TN), False Negative (FN), True Positive (TP), and False Positive (FP).

[3] Correct classification rate or the number of correctly predicted sites divided by the total number of sites.

[4] Mathews Correlation coefficient.

[5] Sensitivity or the number of true positives divided by the total number of sites cleaved.

[6] Specificity or the number of true negatives divided by the total number of sites not cleaved.

[7] Positive Precision or the number of true-positive results divided the total number of positives.

[8] Negative Precision or the number of true-negative results divided by the total number of negatives.

There were 99 incorrect predictions from 40 precursors for at least one model in the Bovine dataset. Of the 17% incorrect predictions in all three models, 71% were false-positive results. The most incorrect predictions (11%) occurred with the Chromogranin A precursor, of which 73% were false-positive results. The remaining precursors accounted for 5% or less of the incorrect predictions and showed no bias due to specific precursors. The majority of the incorrect predictions (32%) occurred in monobasic sites with no basic amino acid in the P2 or P4 locations, most of these occurring with Arg in the P1 location (24% of all incorrect predictions); 63% of them were false positives and many of the remaining incorrect predictions (38%) occurred with a basic amino acid only in the P2 location.

The incorrect predictions from 64 precursors in the Mammalian dataset showed a similar relationship to the Bovine dataset. Of the 862 incorrect predictions, 23% occurred with all models and just 29% occurred with the Properties model. The Chromogranin A precursor had the highest number of incorrect predictions, however, this precursor only accounted for 6% of the incorrect predictions. Monobasic sites and dibasic sites with basic amino acids only present in the P2 and P1 locations accounted for 31% and 44% of the incorrect predictions, respectively.

Across the two datasets, 69% of the incorrect predictions for at least one model were unique predictions in that they only occurred in one window of a precursor from a single species. The remaining incorrect predictions occurred at the same location because the sequences of windows were identical. High sequence homology and non-independence between observations caused an artificial decrease in the predictive performance of the models because the same incorrect prediction was multiplied. Conversely, correct predictions across identical or similar precursor sequences were also observed, thereby artificially increasing the predictive performance of the models.

4 Discussion

The consistency of the cleavage predictions across models in the Mammalian dataset indicated that all models provided highly accurate predictions of cleavage sites in mammalian precursors. The lack of any significant signal or pattern associated with the incorrect predictions indicates that the variability and quality of experimental dataset may be a major factor in incorrect predictions. Determination of model performance is based on the assumption that the experimental data is completely correct, but this assumption may not hold for all cleavage sites and species, especially with less well- studied species or precursors.

False-positive results may be a consequence of a lack of detection, incorrect inference or incorrect inclusion in the datasets. In many species, a number of cleavage sites are likely to remain unidentified because these species have not been as extensively studied as others. For example, both the swine and dog Natriuretic peptide B precursor sequences show additional processing that has not been reported in other species. In addition, some database entries are also likely to be incorrect. For example, the Natriuretic peptide B in the domestic cat appears to have an incorrect cleavage site that is incorrectly assigned to a similar location in the human version as opposed to the more likely correct location in the dog version. Application of the models developed in this study is one strategy to improve the inference of known precursors that lack sufficient experimental validation.

Consideration of the amino acids and properties associated with incorrect cleavage prediction offered insights into approaches to improve model performance. A number of incorrect cleavage predictions occurred in the proximity of the cleavage site and could be assigned to one of two cases, multiple basic sites preceding and following the cleavage site and/or more than one possible cleavage site within the window. In the first case, the processing rules applied to the datasets assumed that cleavage was associated with the most C-terminal basic amino acid when only the bioactive peptide is known. When multiple sequential basic amino acids are present, cleavage can occur between any of these amino acids and still produce the same active peptide because carboxylases will remove any C-terminal basic amino acids. However, without experimental evidence, the actual sequence of the other peptide produced by the cleavage remains unknown. As for the second case, the presence of interconnected cleavage sites with cleavage at one site renders cleavage at other locations impossible. For example, in the Endothelin-1 precursor, the sequence PWRPRRSKRCSCSSLMD is known to be cleaved after the KR site, yet cleavage at the other basic sites results in false-positive predictions. In some cases it is possible that more than one site is cleaved.

The consideration of precursors that may be cleaved by enzymes other than PCs introduces an additional source of bias. For example, the precursor Platelet-Derived Growth Factor D is known to be cleaved by the serine protease Urokinase-type plasminogen activator, which cleaves at the Arg-|-Val bond [14] but is not cleaved by the Furin PC. However, PCs are enzyme candidates that cleave the related Platelet-derived growth factor-B precursor [15].

An additional underlying assumption made when assessing the performance of the models is that all cleavage sites will be cleaved. The basis of this assumption is that different PCs can cleave sequences at the same site and thus, may produce neuropeptides even in the absence of one or more PCs. However, not all PCs are expressed in all tissues and therefore some specific peptides are not formed in all tissues. For example, PC2 is required to process gamma-melanocyte stimulating hormone from the Pro-opiomelanocortin precursor and so this peptide is not detectable in brain regions where PC2 is not expressed. Scamuffa et al. [16] reviewed the effects of PC knockouts in the mouse. Inactivation of Furin or PC5 enzymes results in embryonic mortality at an early age. Mutant mouse lines for PC1, PC2, PC4 and PACE4 mutants are viable, however, they can exhibit retarded growth, hyperproinsulinemia and reduced fertility. Interestingly, PC7 null mice have no apparent defective phenotype. Consequently, a false-positive result could be a true-positive site in another tissue or under other PC combinations, especially for the lesser studied species or tissues.

A unique advantage of using logistic models to predict cleavage in precursors is that the predicted probability of cleavage has an associated measurement of uncertainty or standard error. The uncertainty associated with each predicted probability of cleavage provides indication of the evidence available in the data supporting the prediction. For example, a precursor site may be predicted to be cleaved because the predicted probability of cleavage was higher than the threshold of

0.5 (i.e., 0.55). However, this prediction may have a high confidence interval and thus, the prediction should be weighted by the uncertainty.

The alignment of sequences from many species can also help identify cleavage sites that may be incorrectly predicted. Southey et al. [3] predicted that a single amino acid difference between the human and chimpanzee neuropeptide FF precursor provided a false-positive result since the cleavage only occurred in the human sequence. For example, although Hinuma et al. [17] predicted three human and bovine neuropeptides in the RFamide-related peptide precursor, only two predictions were experimentally validated. The missing predicted neuropeptides did not occur in the rat and mouse sequences and alignment of the sequences from these species showed that the missing sequence was not present in rat and mouse precursors. However, it is possible for a precursor to be differentially expressed between species. For instance, although the Cocaine- and amphetamine-regulated transcript protein precursor is expressed in long (102 amino acids) and short (89 amino acids) forms in mice and rats, only the short form is present in humans [18].

Acknowledgements. This material is based upon work supported by the National Institute on Drug Abuse under Award No. P30 DA018310 to the UIUC Neuroproteomics Center, and the National Institutes of Health under Award No. R01 GM068946.

References

1. Fricker, L.D.: Neuropeptide-processing enzymes: applications for drug discovery. AAPS J. 7 (2005) E449-455
2. Hummon, A.B., Hummon, N.P., Corbin, R.W., Li, L.J., Vilim, F.S., Weiss, K.R., Sweedler, J.V.: From precursor to final peptides: a statistical sequence-based approach to predicting prohormone processing. J. Proteome Res. 2 (2003) 650-656
3. Southey, B.R., Rodriguez-Zas, S.L., Sweedler, J.V.: Prediction of neuropeptide prohormone cleavages with application to RFamides. Peptides 27 (2006) 1087-1098
4. Amare, A., Hummon, A.B., Southey, B.R., Zimmerman, T.A., Rodriguez-Zas, S.L., Sweedler, J.V.: Bridging neuropeptidomics and genomics with bioinformatics: prediction of mammalian neuropeptide prohormone processing. J. Proteome Res. 5 (2006) 1162-1167
5. Southey, B.R., Hummon, A.B., Richmond, T.A., Sweedler, J.V., Rodriguez-Zas, S.L.: Prediction of neuropeptide cleavage sites in insects. Mol Cell Proteomics (2007) (submitted)
6. Southey, B.R., Amare, A., Zimmerman, T.A., Rodriguez-Zas, S.L., Sweedler, J.V.: NeuroPred: a tool to predict cleavage sites in neuropeptide precursors and provide the masses of the resulting peptides. Nucleic Acids Res. 34(Web Server issue) (2006) W267-272
7. Schechter, I., Berger, A.: On the size of the active site in proteases. I. papain. Biochem. Biophys. Res. Commun. 27 (1967) 157-162
8. Berg, J.M., Tymoczko, J.L., Stryer, L.: Biochemistry. 5th edn. WH Freeman and Company, New York (2002)
9. Bendtsen, J.D., Nielsen, H., von Heijne, G., Brunak, S.: Improved prediction of signal peptides: SignalP 3.0. J. Mol. Biol. 340 (2004) 783-795
10. Agresti, A.: An Introduction to Categorical Data Analysis. John Wiley and Sons, New York (1996)

11. Duckert, P., Brunak, S., Blom, N.: Prediction of proprotein convertase cleavage sites. Protein Eng. Des. Sel. 17 (2004) 107-12
12. Baldi, P., Brunak, S., Chauvin. Y., Andersen, C.A., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics 16 (2000) 412-424
13. Matthews, B.W.: Comparison of predicted and observed secondary structure of T4 phage lysozyme. Biochim. Biophys. Acta 405 (1975) 442-451
14. Ustach, C.V., Kim, H.R.C.: Platelet-derived growth factor D is activated by urokinase plasminogen activator in prostate carcinoma cells. Mol. Cell Biol. 25 (2005) 6279-6288
15. Siegfried, G., Basak, A., Prichett-Pejic, W., Scamuffa, N., Ma, L., Benjannet, S., Veinot, J.P., Calvo, F., Seidah, N., Khatib, A.M.: Regulation of the stepwise proteolytic cleavage and secretion of PDGF-B by the proprotein convertases. Oncogene 24 (2005) 6925-6935
16. Scamuffa, N., Calvo, F., Chretien, M., Seidah, N.G., Khatib, A.M.: Proprotein convertases: lessons from knockouts. FASEB J., 20 (2006) 1954-1963
17. Hinuma, S., Shintani, Y., Fukusumi, S., Iijima, N., Matsumoto, Y., Hosoya, M., Fujii, R., Watanabe, T., Kikuchi, K., Terao, Y., Yano, T., Yamamoto, T., Kawamata, Y., Habata, Y., Asada, M., Kitada, C., Kurokawa, T., Onda, H., Nishimura, O., Tanaka, M., Ibata, Y., Fujino. M.: New neuropeptides containing carboxy-terminal RFamide and their receptor in mammals. Nat. Cell Biol. 2 (2000) 703-708
18. Stein, J., Steiner, D.F., Dey, A.: Processing of cocaine- and amphetamine-regulated transcript (CART) precursor proteins by prohormone convertases (PCs) and its implications. Peptides 27 (2006) 1919-1925

Invited Talk: A Computational Study of Bidirectional Promoters in the Human Genome

Mary Qu Yang[1] and Laura L. Elnitski[2]

[1] National Human Genome Research Institute
National Institutes of Health
yangma@mail.nih.gov
[2] Head, Genomic Functional Analysis
National Human Genome Research Institute
National Institutes of Health
elnitski@mail.nih.gov

Abstract. A *bidirectional promoter* is a region along a strand of DNA that regulates the expression of two genes flanking the region. Each of these genes is transcibed in a direction that points away from the other gene; two such genes are said to be in a *head-to-head* configuration. We search the UCSC List of Known Genes and GenBank Expressed Sequence Tag (EST) data for pairs of genes in such a configuration in order to identify new bidirectional promoters.

The EST data constitutes a larger and more intricate dataset than the UCSC List of Known Genes. However, working with EST data presents a challenge, as the EST database may be highly redundant and may also contain overlapping ESTs. To deal with these problems, we have developed an algorithm to identify bidirectional promoters based on the above data sources; the algorithm is capable of handling redundant ESTs, and also ESTs that overlap or disagree in orientation.

This analysis resulted in the identification of thousands of new candidate head-to-head gene pairs, corroborated the 5' ends of many known human genes, revealed new 5' exons of previously characterized genes, and in some cases identified novel genes. Further analyses yielded evidence for coordinate expression of genes in a head-to-head configuration, and examined the prevalence of bidirectional promoters in different biological pathways.

1 Introduction

The mechanisms by which gene expression is regulated in the human genome are as yet not well-understood; it would greatly aid our understanding to be able to pin down prospective regulatory regions. It turns out that candidate regulatory regions can be identified by searching for genes arranged in a "head-to-head" configuration. Recall that a gene has a 5' end and a 3' end; in general, genes are transcribed in the $5' \rightarrow 3'$ direction ("downstream") by an RNA polymerase. The site where the RNA polymerase initially binds is a region of the DNA called a *promoter*; since transcription generally proceeds in the $5' \rightarrow 3'$

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 361–371, 2007.

direction, the promoter must be located upstream of the 5' end of the gene. Two genes that have their 5' ends located fairly close together, say, within 1000 base pairs, and furthermore are transcribed in opposite directions are said to be in a *head-to-head.* configuration. The significance of this configuration is that it is likely that one or more regulatory regions will be located in the stretch between the 5' end of one gene and the 5' end of the other. This stretch is known as a *bidirectional promoter*[1], because it likely serves as the promoter for two genes that are transcribed in opposite directions.

Bidirectional promoters are abundant in the human genome [1], and help to regulate DNA repair, non-DNA housekeeping functions, and other processes. Most early instances of bidirectional promoters were discovered in the course of investigating individual genes [1], but recent work by Trinklein et al. [1] resulted in a substantial increase in the number of known bidirectional promoters.

Spliced Expressed Sequence Tags (ESTs) [2], which are short DNA sequences (usually 200-500 base pairs) obtained by sequencing one or both ends of a transcript of an expressed gene, constitute a large and intricate dataset that can be used to search for bidirectional promoters. However, working with EST data presents a challenge, as the EST database may be highly redundant and may also contain overlapping ESTs. To deal with these problems, we have developed an algorithm to identify bidirectional promoters based on the UCSC List of Known Genes [3], Spliced EST data [2], and GenBank mRNA data [2]; the algorithm is capable of handling redundant ESTs, and also ESTs that overlap or disagree in orientation. The algorithm combines data from the three sources, so that if there is not sufficient evidence to conclude that a candidate region is in fact a bidirectional promoter based on EST data alone, it looks for supporting evidence by examining Known Gene and mRNA data.

This analysis resulted in the identification of thousands of new candidate bidirectional promoters, corroborated the 5' ends of many known human genes, revealed new 5' exons of previously characterized genes, and in some cases identified novel genes. The fact that our algorithm extracts significantly more bidirectional promoters than were previously known raises the question as to whether these are in fact valid promoter regions; we provide supporting evidence to show that this is indeed the case. Further analyses yielded evidence for coordinate expression of genes in a head-to-head configuration, and examined the prevalence of bidirectional promoters in different biological pathways.

2 Data and Algorithm

The data that we use derives from 3 sources:

- The UCSC List of Known Genes [3].
- GenBank mRNA data [2].
- Spliced EST data from the GenBank dbEST database [2].

[1] Here we use the term *candidate bidirectional promoter* to denote a region between two genes in a head-to-head configuration, and the term *bidirectional promoter* to denote such a region that also satisfies the various conditions imposed by the algorithm.

The algorithm for extracting bidirectional promoters is as follows:

I. **Known Gene Analysis:** Known Genes that overlap and have the same orientation are clustered; these clusters are defined by the furthest 3' and 5' ends of any gene in the cluster. The region between the 5' ends of two gene clusters is classified as a bidirectional promoter if the following conditions are satisfied:
   - The 5' ends of the two gene clusters are adjacent to one another, and the two arrows that define the $5' \rightarrow 3'$ direction for each gene cluster point away from each other.
   - The 5' ends of the two gene clusters are separated by no more than 1000 base pairs.
   - There are no other gene clusters between the 5' ends of the two gene clusters.

II. **EST Analysis:** ESTs were assessed for confidence in their orientation using the "ESTOrientInfo" table from the UCSC Genome Browser, which gives a measure of reliability of the orientation of the EST based on all overlapping transcripts from the region. Those with no score were excluded due to low confidence in their orientation. Once the orientation was confirmed, all ESTs were compared to the "intronEST" table to verify agreement; this table lists the intronic orientation for each intron of a spliced EST based on the presence of consensus splice sites.

    ESTs that overlap and have the same orientation are then clustered; these clusters are defined by the furthest 3' and 5' ends of any gene in the cluster. Candidate bidirectional promoter regions are formed by pairing an EST cluster with either another EST cluster or a Known Gene cluster, such that the two clusters are in a head-to-head configuration. The candidate bidirectional promoter is rejected if the two clusters overlap, or if the 5' ends of the two clusters are separated by more than 1000 base pairs.

    The candidate bidirectional promoters are then classified using a decision tree, as shown in Figure 3. The tree either rejects the candidate bidirectional promoter, or assigns it a class label "EST-L*i*", where i is an integer between 1 and 10. To streamline the notation, in the sequel we truncate the leading "EST-L" from the class label, so that the class label is just an integer between 1 and 10. The class label carries two pieces of information:
   - It gives a confidence level that the candidate is in fact a bidirectional promoter. The confidence level is an integer between 1 and 5, where 1 represents the lowest confidence level and 5 the highest. The confidence level can be obtained from the class label via:

     $$\text{confidence level} = 5 - \left\lfloor \frac{\text{class label} - 1}{2} \right\rfloor$$

   - It indicates whether the candidate bidirectional promoter is contained within a Known Gene or not. Odd-numbered class labels indicate that the candidate bidirectional promoter is contained within a Known Gene whereas even-numbered class labels indicate that it is not.

The classification proceeds as follows:

1. Candidate bidirectional promoters enter at the top of the tree in Figure 3. If the candidate bidirectional promoter is contained within a Known Gene, and there exists base pairs of the candidate bidirectional promoter that are more than 1000 base pairs away from the 5' end of the Known Gene in which the candidate bidirectional promoter is contained, then the candidate bidirectional promoter is rejected, otherwise we proceed to Step 2 (the next level of the tree).
2. If the EST cluster(s) flanking the candidate bidirectional promoter satisfy the condition that the number of ESTs that overlap the cluster and disagree in orientation with the cluster is smaller than the number of ESTs comprising the cluster, then we say there is "majority agreement in orientation", and the candidate bidirectional promoter is classified to class 1 or class 2, depending on whether it is contained within a Known Gene. Otherwise we proceed to Step 3 (the next level of the tree).
3. If the EST cluster(s) flanking the candidate bidirectional promoter satisfy the condition that, after disregarding ESTs that disagree in orientation with the cluster and overlap the 5' end of the cluster by no more than 1000 base pairs, the number of ESTs that overlap the cluster and disagree in orientation with the cluster is smaller than the number of ESTs comprising the cluster, then we say there is "majority agreement after excluding 5' overlap", and the candidate bidirectional promoter is classified to class 3 or class 4, depending on whether it is contained within a Known Gene. Otherwise we proceed to Step 4 (the next level of the tree).
4. If the EST cluster(s) flanking the candidate bidirectional promoter agree in orientation with Known Genes and mRNA transcripts, then the candidate bidirectional promoter is classified to class 5 or class 6, depending on whether it is contained within a Known Gene. Otherwise we proceed to Step 5 (the next level of the tree).
5. If the EST cluster(s) flanking the candidate bidirectional promoter agree in orientation with Known Genes, then the candidate bidirectional promoter is classified to class 7 or class 8, depending on whether it is contained within a Known Gene. Otherwise we proceed to Step 6 (the next level of the tree).
6. If the EST cluster(s) flanking the candidate bidirectional promoter exhibit majority agreement (as in Step 2) after excluding ESTs that overlap the 3' end of the cluster, then the candidate bidirectional promoter is classified to class 9 or class 10, depending on whether it is contained within a Known Gene. Otherwise the candidate bidirectional promoter is rejected.

The set of bidirectional promoters extracted by the algorithm consists of those extracted in Step I, which are precisely those flanked by Known Gene clusters, along with those extracted in Step II, which are precisely those flanked either

Table 1. Verification of regulatory regions by TAF250 and CpG overlap

Leaf	Gene Pairs	Valid Taf250 (%)	CpG island (%)	Dual TAF250 (%)
K.G	1,006	74.55	90.15	71.37
EST-L1	2,083	53.77	72.11	50.17
EST-L2	240	50.83	80.41	49.58
EST-L3	225	50.22	73.78	47.11
EST-L4	173	61.85	79.19	58.96
EST-L5	184	37.50	47.80	35.32
EST-L6	103	61.17	67.96	57.28
EST-L7	21	42.86	66.67	33.33
EST-L8	24	29.16	58.33	25.00
EST-L9	363	66.92	83.27	60.84
EST-L10	54	53.70	74.07	48.15
Overall (EST)	3,470	52.30	70.40	48.85

by two EST clusters, or by one EST cluster and one Known Gene cluster, and furthermore are not rejected by the decision tree.

Evidence that the extracted regions indeed serve as promoters can be obtained by looking for two features in the extracted regions that are associated with promoters:

– The presence of experimentally validated TAF250 (or TAF1) binding sites: a high percentage of TAF250 binding sites coincide with other markers of promoter regions [4].
– The presence of CpG islands.

For extracted regions that are flanked by two Known Gene clusters (those extracted in Step I), 74% overlapped a valid TAF250 binding site and 90% overlapped a CpG island, whereas for extracted regions that are flanked either by two EST clusters or by one EST cluster and one Known Gene cluster (those extracted in Step II), 52% overlapped a valid TAF250 binding site and 70% overlapped a CpG island. A summary of the percentages of extracted regions with valid TAF250 binding sites and/or CpG islands for each class is given in Table 1.

Evidence that the extracted regions are in fact *bidirectional* promoters was obtained by dividing the extracted region in half and looking for experimentally validated TAF250 binding sites in each half. The last column of Table 1 gives the percentage of extracted regions with TAF250 binding sites in each half.

3 Results and Discussion

The algorithm identified 1006 bidirectional promoters flanked by two Known Genes, and 159 bidirectional promoters flanked by one Known Gene and one EST cluster (this situation is illustrated in Figure 4(c)). Of 5,575 candidate bidirectional promoters flanked by two EST clusters, 2,105 were rejected by the algorithm. Of the remaining 3,470 identified bidirectional promoters:

- 2,876 were supported by downstream sequences overlapping additional ESTs, mRNA or Known Gene data.
- 594 were located in Known Genes; these alternative promoters direct transcription of both a shorter form of the gene G in which they are embedded and a gene that has the opposite orientation to that of G (this situation is illustrated in Figure 4(b)).

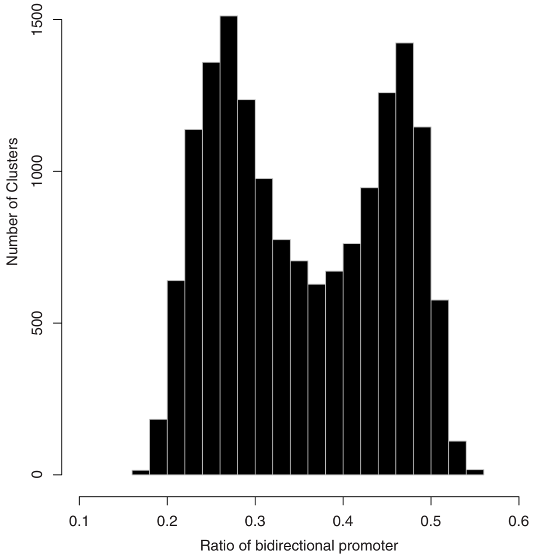


Fig. 1. Histogram of proportions of genes with bidirectional promoters after clustering genes with similar expression profiles

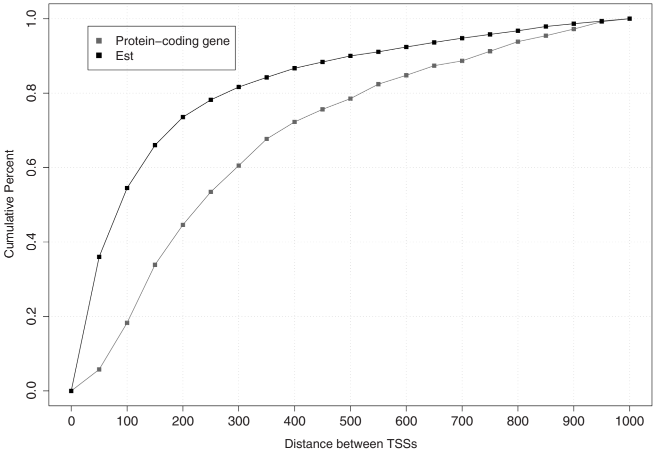


Fig. 2. Percentiles of the widths of the bidirectional promoter regions extracted in the Known Gene analysis and in the EST analysis

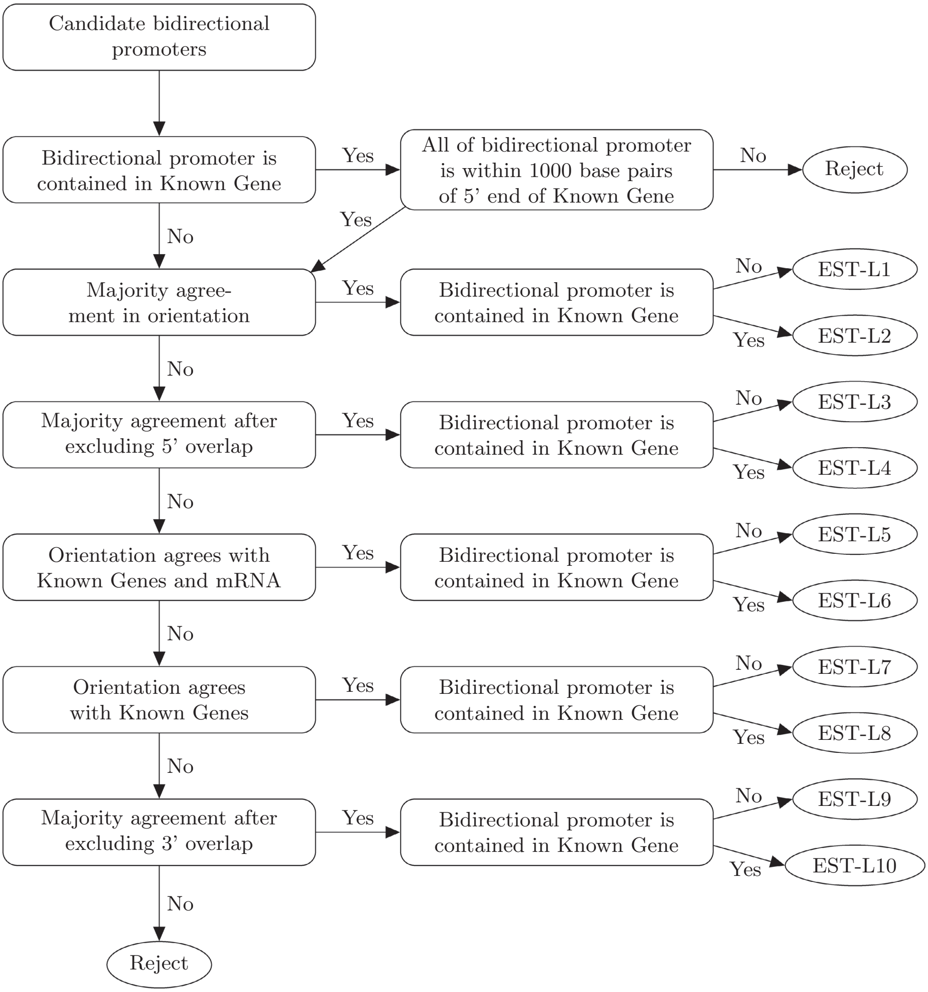

Candidate bidirectional promoters
Bidirectional promoter is contained in Known Gene
Yes
All of bidirectional promoter is within 1000 base pairs of 5' end of Known Gene
No
Reject
Yes
No
Majority agreement in orientation
Yes
Bidirectional promoter is contained in Known Gene
No
EST-L1
Yes
EST-L2
No
Majority agreement after excluding 5' overlap
Yes
Bidirectional promoter is contained in Known Gene
No
EST-L3
Yes
EST-L4
No
Orientation agrees with Known Genes and mRNA
Yes
Bidirectional promoter is contained in Known Gene
No
EST-L5
Yes
EST-L6
No
Orientation agrees with Known Genes
Yes
Bidirectional promoter is contained in Known Gene
No
EST-L7
Yes
EST-L8
No
Majority agreement after excluding 3' overlap
Yes
Bidirectional promoter is contained in Known Gene
No
EST-L9
Yes
EST-L10
No
Reject

Fig. 3. Decision tree for classifying candidate bidirectional promoter regions flanked by at least one EST cluster

3.1 Identification of Novel Genes and Exons

For each Known Gene G that is not in a head-to-head configuration with another Known Gene, let E be the closest EST to G that is in a head-to-head configuration with G. If the 5' end of E is no more than 1000 base pairs away from the 5' end of G, then:

- If E overlaps a downstream Known Gene G_2 having the same orientation as E, then E is considered to be an extension of the 5' end of G_2.
- If E overlaps a downstream gene G_2 having the opposite orientation to E, then E is considered to be a novel gene.

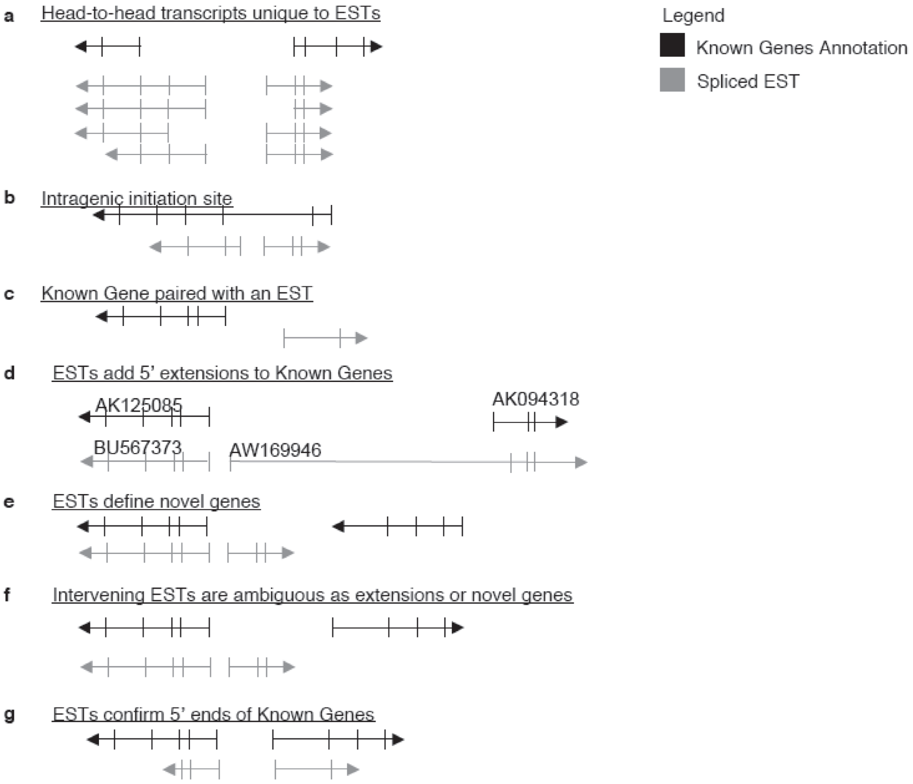

a
Head-to-head transcripts unique to ESTs
b
Intragenic initiation site
c
Known Gene paired with an EST
d
ESTs add 5' extensions to Known Genes
AK125085
AK094318
BU567373
AW169946
e
ESTs define novel genes
f
Intervening ESTs are ambiguous as extensions or novel genes
g
ESTs confirm 5' ends of Known Genes
Legend
Known Genes Annotation
Spliced EST

Fig. 4. Possible configurations of Known Genes and spliced ESTs

– If E does not overlap any Known Gene, but one or more downstream Known Genes have the same orientation as E, then E could either be a 5' extension or a novel gene (this situation is illustrated in Figure 4(f)). These ESTs require further investigation as well as experimental verification to determine if they represent 5' extensions or novel genes.

New functional elements identified in this analysis included novel 5' exons for characterized human genes (this situation is illustrated in Figure 4(d)). For instance, the EST AW169946 extended the 5' end of gene AK094318 by 144,000 base pairs to create a new transcription initiation site adjacent to the neighboring gene, AK125085.

In addition to extension of characterized genes, this analysis identified novel transcripts. These transcripts were absent from the List of Known Gene annotations and therefore were only detected by the EST analysis (this situation is illustrated in Figure 4(e)). These transcripts were spliced, however their protein-coding potential was not always obvious.

Of the 3,470 pairings of EST clusters in a head-to-head configuration, 40% represented extensions of the 5' ends of Known Genes and 43% represented novel

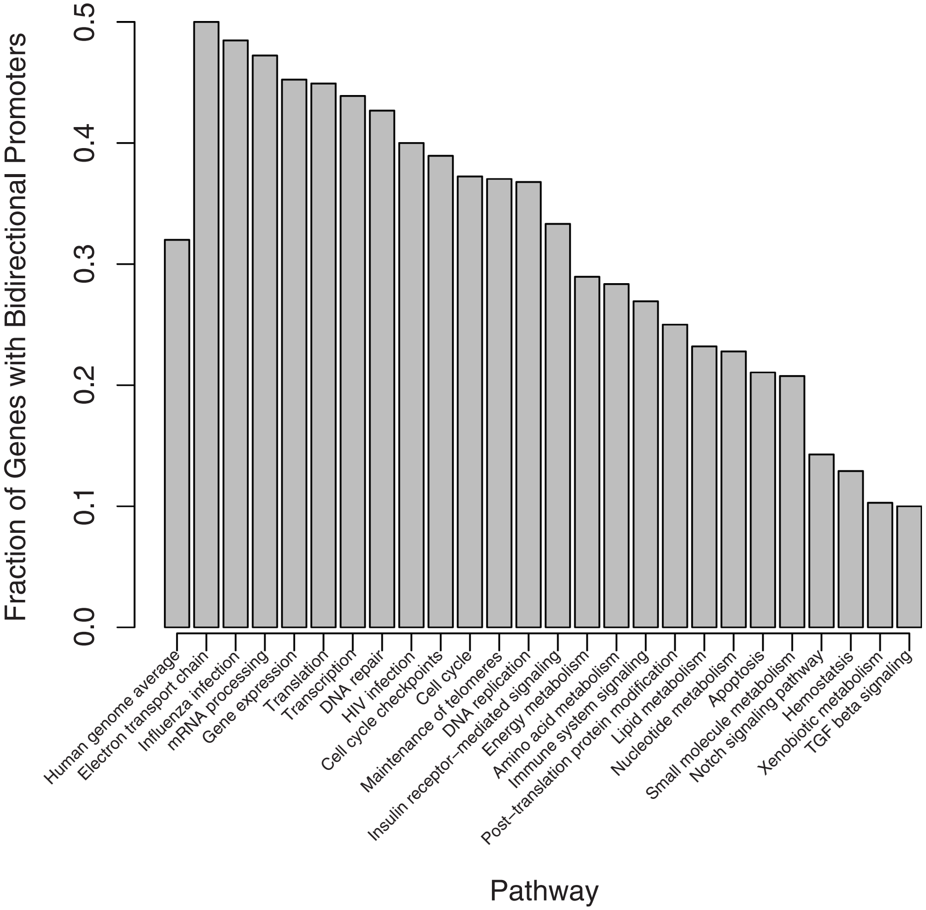


Fig. 5. Fraction of genes regulated by bidirectional promoters for different pathways. The first bar gives the average for the human genome, which is approximately .31.

transcripts. ESTs that confirmed the 5’ ends of Known Genes were abundant (this situation is illustrated in Figure 4(g)).

3.2 Localization of Regulatory Intervals

The abundance of Known Genes whose 5’ ends were extended by the EST analysis indicated that in many cases augmenting the Known Gene data with EST data resulted in narrower, more localized bidirectional promoter regions. To compare the widths of the bidirectional promoter regions extracted in the Known Gene analysis and in the EST analysis, the percentiles of the widths of the bidirectional promoter regions extracted in the Known Gene analysis and in the EST analysis are shown in Figure 2. The curve corresponding to the EST analysis lies below that for the Known Gene analysis, indicating that the EST analysis resulted in narrower, more localized bidirectional promoter regions than the

Known Gene analysis; 80% of the bidirectional promoters identified by the EST analysis were 300 base pairs or less, whereas 80% of the bidirectional promoters identified by the Known Gene analysis were 550 base pairs or less.

3.3 Coordinately-Regulated Expression Groups

We looked for evidence of common regulatory patterns revealed by microarray expression profiles among 16,078 Known Genes. For each Known Gene, a cluster was formed consisting of that Known Gene, along with the 500 Known Genes with the most similar co-expression profiles according to the GNF expression data [5]. The association rate, defined as the proportion of genes in the same cluster that are regulated by bidirectional promoters, was then calculated for each cluster; it ranged from a low of .16 to a high of .56. A histogram of the association rates, shown in Figure 1, reveals a bimodal distribution. Genes with the highest rates clustered with other genes regulated by bidirectional promoters at a ratio of 2:1. The difference between the clusters obtained and those that would be expected by chance was statistically significant. Thus there was strong evidence of coordinated expression among subsets of genes in a head-to-head configuration.

3.4 Prevalence of Bidirectional Promoters in Biological Pathways

Bidirectional promoters are known to regulate a few categories of genes [6,7]. Using the 26 biological pathway genes from the Reactome project [8] we examined additional biological categories for enrichment of bidirectional promoters. Compared to the human genome average in which 31% of genes contained bidirectional promoters, 13 Reactome pathways had a ratio of bidirectional promoters significantly larger than 31%, as shown in Figure 5. For example, the percentage of bidirectional promoters in the Influenza, HIV infection, and DNA repair pathways were respectively 48%, 42%, and 40%; these values yielded respective p-values of 0.04, 0.04, and 0.09 in a Chi-square test, indicating a statistically significant enrichment of bidirectional promoters as compared to the genome average. These results suggest that bidirectional promoters could provide potential therapeutic targets for disease intervention.

Acknowledgements

We gratefully acknowledge discussions with faculty of National Human Genome Research Institute for improvement of this manuscript. This research was supported by the Intramural Research Program of the National Human Genome Research Institute, National Institutes of Health.

References

1. Trinklein, N.D., Aldred, S.F., Hartman, S.J., Schroeder, D.I., Otillar, R.P., Myers, R.M.: An Abundance of Bidirectional Promoters in the Human Genome. Genome Res. **14**(1) (2004) 62–66
2. Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J., Wheeler, D.L.: GenBank: update. Nucl. Acids Res. **32** (2004) D23–26

3. Hsu, F., Kent, W.J., Clawson, H., Kuhn, R.M., Diekhans, M., Haussler, D.: The UCSC Known Genes. Bioinformatics **22**(9) (2006) 1036–1046
4. Kim, T.H., Barrera, L.O., Zheng, M., Qu, C., Singer, M.A., Richmond, T.A., Wu, Y., Green, R.D., Ren, B.: A high-resolution map of active promoters in the human genome. Nature **436** (August 2005) 876–880
5. Su, A.I., Wiltshire, T., Batalov, S., Lapp, H., Ching, K.A., Block, D., Zhang, J., Soden, R., Hayakawa, M., Kreiman, G., Cooke, M.P., Walker, J.R., Hogenesch, J.B.: A gene atlas of the mouse and human protein-encoding transcriptomes. PNAS **101**(16) (2004) 6062–6067
6. Adachi, N., Lieber, M.R.: Bidirectional gene organization: a common architectural feature of the human genome. Cell **109**(7) (June 2002) 807–9
7. Zhao, Q., Wang, J., Levichkin, I.V., Stasinopoulos, S., Ryan, M.T., Hoogenraad, N.J.: A mitochondrial specific stress response in mammalian cells. The EMBO Journal **21** (2002) 4411–19
8. Joshi-Tope, G., Gillespie, M., Vastrik, I., D'Eustachio, P., Schmidt, E., de Bono, B., Jassal, B., Gopinath, G., Wu, G., Matthews, L., Lewis, S., Birney, E., Stein, L.: Reactome: a knowledgebase of biological pathways. Nucl. Acids Res. **33** (2005) D428–432
9. Beissbarth, T., Speed, T.P.: GOstat: find statistically overrepresented Gene Ontologies within a group of genes. Bioinformatics **20**(9) (2004) 1464–1465
10. Chen, C., Gentles, A.J., Jurka, J., Karlin, S.: Genes, pseudogenes, and Alu sequence organization across human chromosomes 21 and 22. PNAS **99**(5) (2002) 2930–2935
11. Cooper, S.J., Trinklein, N.D., Anton, E.D., Nguyen, L., Myers, R.M.: Comprehensive analysis of transcriptional promoter structure and function in 1% of the human genome. Genome Res. **16**(1) (2006) 1–10
12. Trinklein, N.D., Aldred, S.J.F., Saldanha, A.J., Myers, R.M.: Identification and Functional Analysis of Human Transcriptional Promoters. Genome Res. **13**(2) (2003) 308–312

The Identification of Antisense Gene Pairs Through Available Software

Mark J. Lawson and Liqing Zhang

Department of Computer Science, Virginia Tech, McBryde 660,
Blacksburg, VA, 24060
{malawso4,lqzhang}@vt.edu

Abstract. Antisense genes have been shown to have a variety of functions in both prokaryotes and recently in eukaryotes as well. They are hypothesized to be an important part of every genome and have been shown to be evolutionarily conserved as well. Naturally, it is in our interest to develop a software for identifying antisense pairs. While a variety of approaches and software do exist, each approach has its limitations and the software is not meant for large-scale analyses for identifying both cis and trans antisense genes. Here we present a novel way to identify antisense genes and show the results we obtained through it. While in no means a perfect solution, we do manage to show a possible way that may lead to more accurate prediction of antisense genes.

Keywords: Antisense genes, Antisense software, MUMmer.

1 Introduction

1.1 What Are Antisense Pairs?

Antisense pairs (sometimes referred to as NATS: Natural Antisense Transcripts) are pairs of endogenous RNAs in an organism that are complementary to each other. Due to this complementarity, these pairs oftentimes will bind to each other, altering normal processes that these RNA might undergo (such as splicing and translation) and in some cases creating double-stranded RNA. It is also common to refer to antisense pairs as sense/antisense pairs. In this term, the sense RNA is the one that is being affected by the antisense RNA. In cases where only one RNA is protein coding, this coding gene is referred to as the sense RNA, whereas the non-coding RNA is the antisense RNA [1]. In cases where both are coding/non-coding, this assignment is arbitrary, so it makes more sense to use the term “antisense pair” as we shall throughout this paper.

There are two types of antisense pairs: cis and trans. Cis-antisense pairs occur when two RNAs are transcribed from the same locus but on opposing strands. Trans-antisense pairs occur when the RNA is transcribed from different genomic locations. The majority of research has been done on identifying cis-antisense pairs, so the majority of functionality that is described here has only been observed in cis-antisense pairs. However, it is conceivable that trans-antisense pairs have similar effects.

In human and mouse, estimates have placed the amount of transcripts involved in cis-antisense relations between 5 and 10% [2, 3] although recent estimates claim this

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 372–381, 2007.

amount could very well be higher [4] with some claiming it could be as high as over 20% [5]. These amounts seems to be consistent across all analyzed mammals and plants but no estimates for the number of trans-antisense pairs are available [6, 7]. The amount of trans-antisense pairs could easily be as high as cis-antisense pairs because Rosok and Sioud have shown that over 50% of the double-stranded RNAs that they experimentally determined are originated from trans-antisense pairs [8].

Recent research has also shown that cis-antisense pairs are evolutionarily conserved and were twice as likely to remain in their original orientation as non-antisense pairs. This gives further support to antisense pairs being an important part of the genome [9].

1.2 Antisense Functions

Antisense pairs are thought to be responsible for a variety of gene functions, which will be elaborated on in the following paragraphs. All of these functions fall under the main idea of gene regulation. This listing of functions represents only an introduction to antisense pair functions and is in no way meant to be complete. For a more complete review, we encourage the reader to read several of the reviews we have annotated in our reference list [1, 10-12].

A function that can be observed in cis-antisense pairs is transcriptional interference. For instance, it has been shown that the α1(I) collagen gene produces low-levels of mRNA in the chick embryo chondrocytes [13]. This is due to an antisense gene being located directly across from it, which has increased transcription and seems to prevent the transcription of the sense gene. In a study using *Saccharomyces cerevisiae* [14], Prescott and Proudfoot found that when the *GAL10* and *GAL7* genes were placed on opposite strands in convergent gene orientation, both genes are transcribed at full levels. But as soon as the two genes overlap, the amount of mRNA they produce drops.

A very common function of antisense pairs is that of RNA masking. This occurs when two RNA strands bind to each other which mask the key regulatory features that are responsible for further steps in gene expression such as splicing, transport, translation and degradation. For instance, the Rev-ErbAα gene in rat is responsible for preventing the normal splicing of the ErbAα gene, a thyroid hormone receptor, in rat. The antisense transcript of Rev-ErbAα binds with the ErbAα2 transcript, which prevents its splicing and thereby its normal gene expression as well [15, 16].

A specific case of gene regulation can be seen in X-inactivation [17]. This is partially achieved through the expression of the *Xist* gene. The *Xist* gene possesses a cis-antisense gene called *Tsix* that is expressed equally before the onset of X-inactivation in an organism. After X-inactivation, it seems to disappear, creating no more transcription.

Further examples of antisense functions stem from interactions that occur due to double-stranded RNA. Two of these interactions are RNA editing and RNA interference. RNA editing reactions alter an RNA strand by changing one or two nucleotides in it [18], which in turn can have an effect on coding and the RNA structure, among other things. Peters et al. observed an “A-to-G” (adenine is converted to guanine) edit that was caused by antisense genes in the *Drosophila* *melanogaster* genome [19]. The two genes *4f-rnp* and *sas-10* create RNA that bind together, causing the "A-to-G" edit to occur in both RNA strands.

RNA interference refers to double-stranded RNA that prevents the creation of gene products that are homologous to the double-stranded RNA, causing gene silencing. This double-stranded RNA can be introduced into eukaryotes to prevent the production of a specific gene [20], but the effect has also been observed to happen within organisms themselves through antisense pairs. The *Stellate* genes have been known to undergo gene silencing in the testes of *Drosophila melanogaster* through this process [21].

One aspect of antisense pairs that seems to require further investigation is the role antisense pairs play in genetic disease. A few examples have been discovered [10]. One example can be found in the antisense transcript that reduces the expression of the *UBE3A* gene [22]. This can in turn lead to Prader-Willi syndrome and Angelman syndrome. Antisense pairs are also thought to play a role in DNA methylation and gene silencing leading to diseases such as anemia [23].

1.3 Antisense Software and Computational Methods

With so much discovered functionality and with so much more to be discovered (especially in the field of associated human genetics diseases), it is not surprising that much research has been done on the identification of these antisense pairs. While some have a biological approach of doing this [8], others generally use more computational methods to identify them, as this can be done with ease on a large scale. Despite the fact that many analyses have been done on identifying antisense genes, no research group has come up with a program that can be used for large-scale antisense pair identification. Most just have online databases of their results with an interface that allows the user to search for possible antisense pairs to the given sequence ID. While this is a useful tool for researchers, the majority of these databases are not updated as newer UniGene builds are released and EST databases are updated.

Most approaches that have been done rely on the use of ESTs and/or mRNA to identify antisense pairs. One basic idea is to locate expressed areas of the genome that overlap and lie on opposite strands. This approach is used by Yelin et al. in their "Antisensor" algorithm [2]. It consists of determining the orientation of ESTs through poly(A) tails and signals and then mapping these oriented ESTs to genomes and thereby identifying genes that are likely cis-antisense pairs. They provide an online search tool to go through their results but no downloadable software is available.

Kiyosawa et al. [3] used a combination of cDNA and mRNA to determine overlapping genes. They used cDNA from the FANTOM2 project and mRNA obtained from GenBank to determine 2481 sense-antisense pairs in mouse. They also provide a web interface to view their results. The same analysis was later done with the FANTOM3 dataset [4].

A recent addition to the list of online databases is the NATsDB [24, 25]. Using the UniGene ESTs and the UCSC genome build to map expressed regions to genomes, Zhang et al. have identified cis-antisense pairs in ten different organisms (including human and mouse). Furthermore, they use a "pipeline" approach to update their results as new build of UniGene are available.

While the previous approaches seem to have relatively accurate datasets, they do suffer from the fact that they only identify cis-antisense pairs. Furthermore, due to the very nature of their analyses, it seems unlikely that one would be able to extend these methods to search for trans-antisense pairs. So a different approach seems warranted.

One such approach can be seen in the AntiHunter program [26, 27]. It is an online search tool that uses a given sequence to conduct a search for possible antisense transcripts in an EST database. They have a complete EST database for a variety of organisms and use RepeatMasker and a BLASTN search to find the antisense pairs. While their tool can be used to identify antisense pairs, the interface is cumbersome and slow (results are e-mailed and can take several hours for long sequences) and it has been reported that it presents a large amount of false positives [11]. The online interface only allows for one sequence at a time, making it less than ideal for large-scale analysis.

An approach that has been used to identify trans-antisense pairs was presented in a paper by Wang et al. [7]. In their approach Wang et al. used BLAST on aligned *Arabidopsis thaliana* cDNA and identified sequence complementarity between the sequences. They came up with two levels of complementarity for a pair of cDNAs: "high coverage" when at least one cDNA has at least 50% of their sequence complement to the other sequence and "100 nt" when a chunk of at least 100 contiguous nucleotides was identified between the two pairs. These results were then validated by the use of a hybridization modeling program called hybrid [28, 29] that showed that 100% of the high coverage antisense pairs would hybridize and 90% of the "100 nt" antisense pairs would hybridize as well. These results were then clustered and evaluated along many metrics such as gene expression and gene ontology.

Our future goal is to develop a robust program for identifying both trans- and cis-antisense pairs for large scale analyses. The idea is that the program does not require the use of EST databases but can determine antisense pairs through sequence alone. We then hope to distribute this program to aid the research community in the search for antisense pairs. Furthermore, we want to be able to create a database of antisense pairs with an online tool that allows the user to search for antisense transcripts based on a given gene ID. We plan to setup antisense databases for every annotated genome.

To test the method we have developed, we decided to test the hypothesis set forth by Kiyosawa et al. [3]. Through their analysis of cis-antisense pairs on the mouse genome they came to the hypothesis that the X chromosome has less antisense pairs than other chromosomes, possibly due to X-inactivation. This hypothesis was supported by Zhang et al. [25] who determined that in mammals this holds for the cis-antisense pairs they determined. In this study we present our preliminary studies on identifying antisense genes in human chromosome X and chromosome 10 (chosen because they possess a similar amount of genes), using MUMmer and rules similar to those used by Wang et al. [7] to see if the hypothesis holds true for trans-antisense pairs as well.

2 Materials and Methods

2.1 MUMmer and Input Data

In an attempt to identify a novel method to identify antisense pairs in a genome, we chose to use the program MUMmer [30]. MUMmer is an efficient pattern matching tool that uses a suffix tree approach to identify maximal unique matches between sequences. The way this is done is the program first builds a suffix tree out of the reference sequences and then compares each of the query sequences to this suffix tree, identifying the longest unique matches for both the reference and the query sequence.

For our dataset we used all annotated human transcripts from the ENSEMBL database [31] (version 41). We limited ourselves to only those annotated to a chromosome but included all biotypes not just "protein coding" transcripts to capture all annotated non-coding RNAs as well. Furthermore, we filtered out all transcripts that possessed unknown nucleotides (denoted with an "N") as they proved to obscure the MUMmer data. The remaining transcripts were stored in fasta-formatted files, organized by chromosome (due to its size, chromosome 1 was split into two files).

The first step consisted of creating reverse complements to these transcripts. This was done easily with a perl script that complemented each nucleotide and reversed the sequence. MUMmer was then run by using each chromosome as the reference sequence and then using the complements of this chromosome and all other chromosomes as query sequences. The sequences had to be split up as even just loading chromosome 1 into a suffix tree would cause MUMmer to segment fault on our server. While no complete time analysis was done, MUMmer ran quite efficiently, taking on average an hour to run for each chromosome. The MUMmer results were then parsed with the use of perl scripts and inserted into a MySQL database to eliminate duplicated entries and to link the transcripts with their corresponding genes.

2.2 Antisense Quantification

To quantify our data we chose a method similar to Wang et al. [7]. We basically split our generated MUMmer data into two levels of pairs, those with "high coverage" and those with a contiguous matching region of at least 100 nt. The "high coverage" antisense pairs (or "HC" antisense pairs as we shall henceforth refer to them) are pairs in which at least 50% of one transcript is covered by the matching region. This is different from the way Wang et al. classified this as they do not use a contiguous region as we do. Our way is more stringent. We chose this method because through their hybridization tests, Wang et al. had an accuracy rate of 100% with the HC pairs and 90% with the "100 nt" pairs.

3 Results

In total, 977 genes on chromosome X with 2702 transcripts and 904 genes on chromosome 10 with 2805 transcripts were analyzed (after filtering). Out of these transcripts, 741 form either an HC or 100 nt antisense pair on chromosome X and 729 form either an HC or 100 nt antisense pair on chromosome 10. This represents 27.4% and 26% of their respective transcripts. Based on this, we cannot confirm the hypothesis stated by Kiyosawa et al. based on the fact that not only does the X-chromosome have more than 20% of its transcripts form antisense pairs, but more antisense paired transcripts were found in chromosome X than in chromosome 10.

The results of linking these transcripts to their respective genes can be seen in Table 1. As can be seen on first glance, there exist more antisense pairs in chromosome 10 for both groups (HC and 100 nt), despite the fact that there are slightly less genes in the chromosome. In total, if we just look at the genes within the chromosome, 94 genes possess an HC antisense partner, 194 possess a 100 nt partner and 262 have either one or the other on chromosome X. This represents 9.6%, 19.8%, and 26.8% of all the genes on the chromosome. If we include trans-antisense pairs that have genes from other chromosomes, the amount of genes on chromosome X increase to 98 (10%) for HC, 233 (23.8%) for 100 nt, and 297 (30.4%) in total.

For chromosome 10, if we focus only on pairs where both genes of the antisense pair are on the same chromosome, we find 112 genes (12.3%) with an HC partner, 174 (19.2%) with a 100 nt partner, and 254 (28.1%) have either one or the other. Expanding our results to antisense pairs that include genes from other chromosomes, there are 114 genes (12.6%) with HC antisense pairs, 204 (22.5%) with 100 nt, and 280 (31%) total.

Evaluating the gene amounts, we can see that like the transcript amounts mentioned earlier, there is no evidence to confirm the Kiyosawa et al. hypothesis. The amount of genes that form antisense pair is percentage wise close to identical with minimal deviations. A difference does however lie in the amount of cis-antisense pairs we have found. A total of 145 cis-antisense paired genes were found on chromosome 10 (16% of all genes) while only 120 were found on chromosome X (12.3%). This does give some support to the hypothesis, especially since the focus of the hypothesis is on the amount of cis-antisense pairs the chromosome possesses.

Looking further at Table 1 we can see more interesting trends. For instance, for both chromosomes the majority of HC antisense pairs involve genes that are both on the respective chromosome (i.e. both genes lie on the X chromosome). Also, the majority of these pairs are cis-pairs in both chromosomes. If we look at the 100 nt pairs, we see that the majority of trans-antisense pairs occur with genes on other chromosomes (i.e. a gene on chromosome X forms an antisense pair with a gene on a different chromosome) and that less cis-pairs exist.

A noticeable difference is in the amount of convergent and divergent cis-pairs. In chromosome X there are more divergent cis-pairs as opposed to convergent pairs. However, in chromosome 10 divergent and convergent cis-antisense pairs are practically equal.

Another trend we noticed in our analysis is that the genes that form an antisense pair (both HC and 100 nt) have more transcripts associated with them (i.e. are more likely to be alternatively spliced) than the norm. In chromosome X the average gene has 2.77 transcripts associated with it, while in chromosome 10 this number is 3.1. However the antisense genes have an average transcript amount of 3.44 and 3.67 for chromosome X and 10, respectively. Based on this, it would seem that genes with more alternative splicing are slightly more likely to form antisense pairs.

In figure 1 we compared how many transcripts genes that are a part of an antisense pair and genes that are not a part of an antisense pair possess. We did this by looking at the percentage of all the genes that possess a transcript of a certain number. As can be seen on both graphs, non-antisense genes have more genes with less transcripts (1 and 2) and antisense genes possess in general more transcripts (> 2).

Table 1. Table of antisense gene data

	Chromosome X		Chromosome 10	
Total Genes		977		904
	HC	100 nt	HC	100 nt
Convergent cis pairs	15	6	29	11
Divergent cis pairs	32	11	29	12
Same chromosome trans	6	379	2	322
Other chromosome trans	5	5732	6	6826
Total	58	6128	66	7171

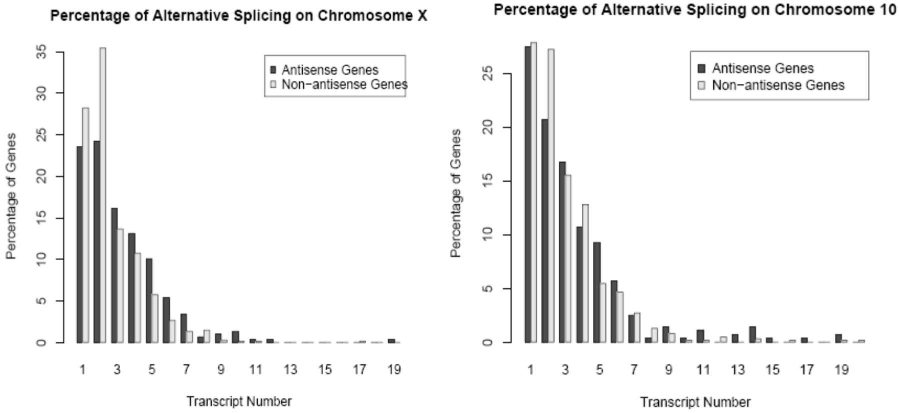


Fig. 1. Percentage of alternate splicing on both chromosomes (i.e. percentage of genes that have a certain amount of transcrips associated with them

In figure 2, we graphed the percentage of genes on other chromosomes. We totaled up the amount of genes involved in all antisense pairs (whether HC or 100 nt) and determined what percent of those genes came from each chromosome. Interesting to see here, is the fact that percentage wise there is little to no difference in terms of where the antisense genes come from.

A further analysis was carried out to determine the biotypes of these antisense pairs. On both chromosomes the majority of pairs consisted of transcripts that were both "protein-coding." However a small amount did exist in which one transcript was "protein-coding" and one was deemed a "pseudogene" (alternative biotypes were not discovered). On the X-chromosome 3 pseudogenes were discovered that were a part of an antisense pair with another gene (in all cases, all of their transcripts were complementary as well). Two of these pseudogenes were part of HC cis-antisense pairs and the other one formed a 100 nt complement with the *TBL1Y* gene on the Y chromosome.

Chromosome 10 was found to have more antisense pairs that involved pseudogenes. Specifically, 12 pseudogenes were found to be involved in antisense pairs of which 6 can be found on chromosome 10 and the remaining on other chromosomes (1, 20, and

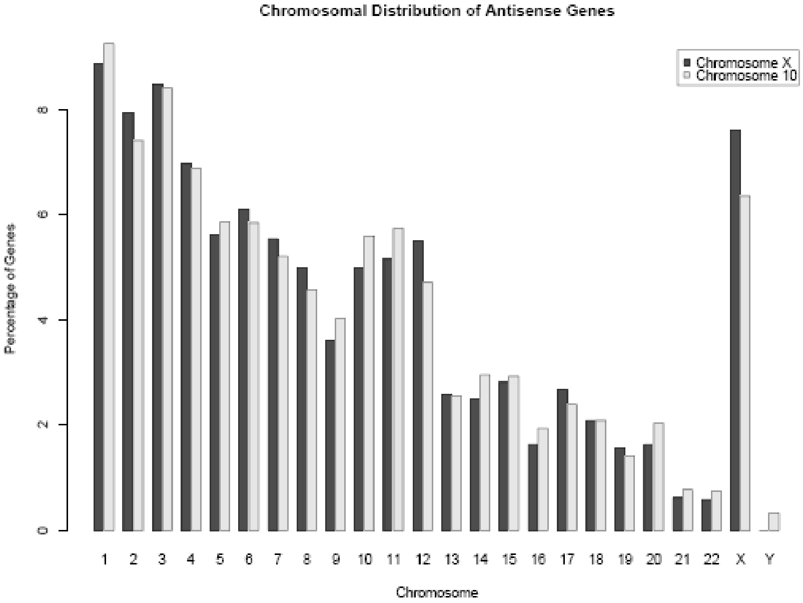


Fig. 2. Percentage of antisense genes that are derived from each chromosome

21). Of the pseudogenes on chromosome 10, 3 are a part of cis-antisense pairs, 2 form trans antisense pairs that are still within this chromosome, and 1 forms antisense pairs with a genes located on chromosome 1 and 4. Of note among these pseudogenes is the gene "ENSG00000203294" (no additional name is given). Not only does it form an HC cis-antisense pair with the *ABCC2* gene, it also forms an antisense pair with 5 other genes (2 on chromosome 1 and 1 each on chromosomes 5, 8, and 12). Of interest here is that these so-called pseudogenes may still play an important role in the regulation of gene expression.

4 Discussion

Overall our analysis gives only marginal support to the Kiyosawa et al. hypothesis and shows that the inclusion of trans-antisense pairs may in fact call this hypothesis into question. We did see that there are more cis-antisense gene pairs in chromosome 10 but when including trans-antisense pairs there are only slightly more antisense pairs in chromosome 10. Based on our results, we certainly do not see support for the hypothesis that the X-chromosome is "under represented" when it comes to antisense pairs.

There is however a variety of problems with the results we present here. While we can feel relatively confident that at least the HC genes are very likely to form antisense pairs (and that a "good chunk" of the 100 nt pairs will as well), our results are most likely too narrow. While MUMmer presents a good and efficient way to locate exact matches, it does not identify inexact matches, due to the nature of using a suffix tree. In order for us to have better results, we would need to use or develop an algorithm that takes into account possible mismatches that will not prevent the two strands from binding. So it would be unwise for us to reject or accept the hypothesis based on our obtained data. It merely suggests that further study into this may be required.

A question that was brought up by Rosok and Sioud [8, 32] was whether BLAST identified complementary sequences would even bind in an organism. The fact that two sequences possess complementary sequences (and even hybridization programs predict they would bind) does not necessarily mean they would actually bind in an organism. What if they never come in contact with each other? At best, we can now present good “guesses” as to what antisense pairs might exist.

What we propose to do in future work is to develop an algorithm that uses a dataset of known antisense pairs to identify other antisense pairs. Through a machine learning approach such as a Hidden Markov Model or the use of SVMs, we could train the identification of antisense pairs based on parameters such as sequence and genomic location. We could even do organism specific training as more datasets become available, so as to incorporate data specific to each organism as well.

The goal is also to develop software that is not dependent on current gene annotation and able to perform large-scale antisense gene identification. Identifying genes in a genome is a “guessing game” in some cases, so our software will take in entire chromosomal strands and identify regions that may likely form antisense pairs. Thus it can identify antisense regions that may not be currently annotated. Meanwhile, we hope to be able to show the results in an online database and make the software package available for download.

References

1. Munroe, S.H. and J. Zhu, Overlapping transcripts, double-stranded RNA and antisense regulation: A genomic perspective. Cellular and Molecular Life Sciences, 2006. 63(18): p. 2102-2118.
2. Yelin, R., et al., Widespread occurrence of antisense transcription in the human genome. Nature Biotechnology, 2003. 21(4): p. 379-386.
3. Kiyosawa, H., et al., Antisense transcripts with FANTOM2 clone set and their implications for gene regulation. Genome Research, 2003. 13(6B): p. 1324-1334.
4. Katayama, S., et al., Antisense transcription in the mammalian transcriptome. Science, 2005. 309(5740): p. 1564-1566.
5. Chen, J.J., et al., Over 20% of human transcripts might form sense-antisense pairs. Nucleic Acids Research, 2004. 32(16): p. 4812-4820.
6. Wang, X.J., T. Gaasterland, and N.H. Chua, Genome-wide prediction and identification of cis-natural antisense transcripts in Arabidopsis thaliana. Genome Biology, 2005. 6(4): p. -.
7. Wang, H., N.-H. Chua, and X.-J. Wang, Prediction of trans-antisense transcripts in Arabidopsis thaliana. Genome Biology, 2006. 7(10): p. R92.
8. Rosok, O. and M. Sioud, Systematic identification of sense-antisense transcripts in mammalian cells. Nature Biotechnology, 2004. 22(1): p. 104-108.
9. Dahary, D., O. Elroy-Stein, and R. Sorek, Naturally occurring antisense: Transcriptional leakage or real overlap? Genome Research, 2005. 15(3): p. 364-368.
10. Lavorgna, G., et al., In search of antisense. Trends in Biochemical Sciences, 2004. 29(2): p. 88-94.
11. Makalowska, I., C.F. Lin, and W. Makalowski, Overlapping genes in vertebrate genomes. Computational Biology and Chemistry, 2005. 29(1): p. 1-12.

12. Knee, R. and P.R. Murphy, Regulation of gene expression by natural antisense RNA transcripts. Neurochemistry International, 1997. 31(3): p. 379-392.
13. Farrell, C.M. and L.N. Lukens, Naturally-Occurring Antisense Transcripts Are Present in Chick-Embryo Chondrocytes Simultaneously with the down-Regulation of the Alpha-1(I) Collagen Gene. Journal of Biological Chemistry, 1995. 270(7): p. 3400-3408.
14. Prescott, E.M. and N.J. Proudfoot, Transcriptional collision between convergent genes in budding yeast. Proceedings of the National Academy of Sciences of the United States of America, 2002. 99(13): p. 8796-8801.
15. Munroe, S.H. and M.A. Lazar, Inhibition of C-Erba Messenger-Rna Splicing by a Naturally-Occurring Antisense Rna. Journal of Biological Chemistry, 1991. 266(33): p. 22083-22086.
16. Hastings, M.L., et al., Post-transcriptional regulation of thyroid hormone receptor expression by cis-acting sequences and a naturally occurring antisense RNA. Journal of Biological Chemistry, 2000. 275(15): p. 11507-11513.
17. Lee, J.T., L.S. Davidow, and D. Warshawsky, Tsix, a gene antisense to Xist at the X-inactivation centre. Nature Genetics, 1999. 21(4): p. 400-404.
18. Bass, B.L., RNA editing by adenosine deaminases that act on RNA. Annual Review of Biochemistry, 2002. 71: p. 817-846.
19. Peters, N.T., et al., RNA editing and regulation of Drosophila 4f-rnp expression by sas-10 antisense readthrough mRNA transcripts. Rna-a Publication of the Rna Society, 2003. 9(6): p. 698-710.
20. Hannon, G.J., RNA interference. Nature, 2002. 418(6894): p. 244-251.
21. Aravin, A.A., et al., Double-stranded RNA-mediated silencing of genomic tandem repeats and transposable elements in the D-melanogaster germline. Current Biology, 2001. 11(13): p. 1017-1027.
22. Runte, M., et al., The IC-SNURF-SNRPN transcript serves as a host for multiple small nucleolar RNA species and as an antisense RNA for UBE3A. Human Molecular Genetics, 2001. 10(23): p. 2687-2700.
23. Tufarelli, C., et al., Transcription of antisense RNA leading to gene silencing and methylation as a novel cause of human genetic disease. Nature Genetics, 2003. 34(2): p. 157-165.
24. Zhang, Y., et al., NATsDB: Natural Antisense Transcripts DataBase. Nucl. Acids Res., 2007. 35(suppl_1): p. D156-161.
25. Zhang, Y., et al., Genome-wide in silico identification and analysis of cis natural antisense transcripts (cis-NATs) in ten species. Nucleic Acids Research, 2006. 34(12): p. 3465-3475.
26. Lavorgna, G., et al., AntiHunter: searching BLAST output for EST antisense transcripts. Bioinformatics, 2004. 20(4): p. 583-585.
27. Lavorgna, G., et al., AntiHunter 2.0: increased speed and sensitivity in searching BLAST output for EST antisense transcripts. Nucleic Acids Research, 2005. 33: p. W665-W668.
28. Markham, N.R. and M. Zuker, DINAMelt web server for nucleic acid melting prediction. Nucleic Acids Research, 2005. 33: p. W577-W581.
29. Dimitrov, R.A. and M. Zuker, Prediction of hybridization and melting for double-stranded nucleic acids. Biophysical Journal, 2004. 87(1): p. 215-226.
30. Kurtz, S., et al., Versatile and open software for comparing large genomes. Genome Biology, 2004. 5(2): p. -.
31. Ensembl Genome Browser. [cited 2006 12/20]; Available from: http://www.ensembl.org.
32. Rosok, O. and M. Sioud, Systematic search for natural antisense transcripts in eukaryotes - (Review). International Journal of Molecular Medicine, 2005. 15(2): p. 197-203.

Inferring Weak Adaptations and Selection Biases in Proteins from Composition and Substitution Matrices

Steinar Thorvaldsen[1], Elinor Ytterstad[2], and Tor Flå[2]

[1] Tromsø University College, AFL-Informatics, 9293 Tromsø - Norway
[2] Dept of Mathematics and Statistics, University of Tromsø, 9037 Tromsø - Norway
steinar@hitos.no

Abstract. There is a desire for increasing use of statistical methods in analysing the growing amounts of bio-sequences. We present statistical methods that are useful when a protein alignment can be divided into two groups based on known features or traits. The approach is based on stratification of the data, and to show the applicability of the methods we present analysis of genomic data from proteobacteria orders. A dataset of 25 periplasmic/extracellular bacterial enzyme *endonuclease I* proteins was compiled to identify genotypic characteristics that separate the cold adapted proteins from ortholog sequences with a higher optimal growth temperature. Our results reveal that the cold adapted protein has a significantly more positively charged exterior. Life in a cold climate seems to be enabled by many minor structural modifications rather than a particular amino acid substitution. Redistribution of charge might be one of the most important signatures for cold adaptation.

Keywords: Stratified data, Two-way ANOVA, Mantel-Haenszel test, cold adaptation.

1 Introduction

The polar biological sciences stand on the threshold of a new epoch, because of the availability of proteins and genomes of their constituent organisms [1]. The years 2007/08 are announced as international polar years. For both the Northern and Southern polar region, great distances, physical isolation, long periods of darkness, and extreme climates have always posed special challenges. Extreme environments are in general those that fall outside the limited range in which we and most other mesophilic organisms can survive, and are populated by the *extremophiles*. Among extremophiles, which include thermophiles, psychrophiles, barophiles, halophiles and acidophiles, those which live and prefer low temperatures are the largest but still one of the least studied groups. The major proportion of biomass on earth is generated by microorganisms in the world's oceans at cold temperatures (≤ 5°C), and the *psychrophilic* organisms live only in this type of permanently cold habitats, often close to the freezing point of water. All macromolecules of such organisms must be stable and functional in the temperature range in which the species lives, and it is of great interest to understand how this is achieved [2].

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 382–393, 2007.

Living at low temperatures requires a multiplicity of crucial adaptations including maintenance of enzymatic activities at appropriate levels, since this is decreasing exponentially with decreasing temperature. At these temperatures a number of environmental factors are also changed; the solubility of gases is not the same, the viscosity of water increases severalfold as temperature is changed towards the extreme areas, for example.

The present lack of consensus among studies [2] has given rise to the recognition that no set of simple factors distinguish all mesophile and psychrophile proteins. If there are no general rules to cold adaptation and functionality, a more specific approach to the problem will be required to elucidate them. This accentuates the importance of protein- and taxon-specific comparisons. In particular, we try to identify trends in amino acid composition, substitutions and structural features of proteins, which distinguish specific proteins of cold-adapted bacteria from their mesophilic counterparts.

In the present study, we focus on one protein from a relatively narrow range of closely related species belonging to the group of marine gamma and delta *proteobacteria* - a strategy also adopted by [3]. Alignment-free analysis has been used previously to compare amino acid compositions in whole genome and proteome datasets [4, 5]. In our comparative study, we in addition employ alignment-based methods for examination of significant differences between *mesophilic* (*M*) and *psychrophilic* (*P*) groups.

Several articles concerning sequence comparisons, which aim to find common denominators for cold adaptation, have been published in recent years [2]. There is still no general consensus in the field, and different groups of proteins may adapt in different ways. To study closer the mechanisms involved in protein cold adaptation on a molecular level, the enzyme *endonuclease I* has been chosen from related mesophile and psychrophile bacteria [6, 7]. *Endonuclease I* is a periplasmic or extracellular, monomeric enzyme known to cleave both RNA and DNA in a sequence-independent manner, at internal sites. By using a bioinformatics approach, we attempt to find the trends in temperature adaptations of this enzyme.

2 Materials and Methods

2.1 Sequence Data

Orthologues of bacterial *endonuclease I* protein sequences from marine organisms belonging to the gamma and delta subdivision of proteobacteria were gathered from various genome sequence projects around the world, and a total of 48 amino acid sequences were found, when only one sequence per bacterial species was considered. Often the homologues are so distantly related that the amino acid changes due to temperature adaptation are indistinguishable from those originating from random genetic drift, and adaptation to other factors than temperature. Therefore, plasmid sequences and sequences which showed abnormal phylogenetic placement were discarded from the analyses as were sequences suspected to be horizontally transferred or recombined.

The relative GC% of the nucleases suspected to be horizontally transferred was compared with the corresponding average GC of the genome. The data for genomic GC was found in literature, and GC% of the nuclease genes was calculated by the program BioEdit [8]. Sequences suspect of horizontal transfer, or from organisms with extraordinary low GC%, were discarded from the analysis due to the fact that low GC% also affects the codon usage and hence the amino acid preferences [9]. Prediction of the signal-sequence cleavage site was performed using the SignalP-web-server [10], and the signal-sequences were removed.

The optimal growth temperature, T_{opt}, was found by studying the literature, and by searching The Prokaryotic Growth Temperature Database, PGTdb [11]. The sequences were sorted according to phylogeny, and bacterial families with both psychrophilic and mesophilic members were studied in particular [12]. *Endonuclease I* is most similar between the gamma proteobacteria orders *Vibrionales* and *Alteromonadales*, and the delta proteobacteria order *Desulfobacterales*. We made a special study of these three groups, with sequence identity in the range from 30 to 97%, and sequence length from 194 (*Pseudoalteromonas tunicata*) to 233 (*Desulfutalea psychrophilia*) amino acids. Of the original 48 sequences, 25 were now left in our material for further analyses. On these sequences we conducted the statistical tests. They are assembled from *three taxonomic orders,* and from *two growth groups* defined by temperature adaptation: mesophilic (17 sequences), and psycrophilic (8 seq.). Temperature is the main environmental trait that separates these groups, not factors like highly concentrated salts or toxics.

A structural based alignment of the remaining 25 sequences was created using the crystal structure of *V. vulnificus* nuclease (*Vvn*) as guide [7]. Furthermore, for the sake of a more specific analysis, the solvent Accessible Surface Area (ASA) was calculated using the program GETAREA [13] with the crystal structure of *Vvn* (PDB id: 1OUO) as template with default settings. The spatial location was attached according to solvent accessible surface area of the side-chain, where core is defined as 0-9% exposed side-chain, twilight zone 9-36%, and surface 36-100%. With these cut-offs the number of amino acids is distributed approximately equally between the three bins. This method makes it possible to analyse the data relative to some of its 3D structural location. We looked for patterns in the data following from this partitioning and decompositions. The secondary structure was also determined from *Vvn*.

Our main goal is to detect potential differences between the sequences, not to examine the particular organisation and conservation of the enzyme in the study. Therefore, the sites in the alignment with no difference (the conserved sites) were discarded from the analysis. The dataset in general will be unbalanced, in the sense that there are different numbers of sequences in each group.

2.2 Amino Acid Composition

Data consists of 25 aligned amino acid sequences from two groups, 17 mesophilic (*M*) and 8 psycrophilic (*P*). There are altogether 20 different amino acids, and we may consider the amino acid occupying a particular site along the sequence, a random categorical variable X with possible values {A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T, W,Y,V}.

For convenience the amino acids are often encoded by numbers $x = 1, 2,\ldots, 20$. A sequence may contain gaps, which are given the value $x = 21$.

The amino acids may also be joined in many ways based on common characteristics, and hence the amino acid alphabet can be reduced. Based on charge distribution, one may divide them into four main categories: *negatively charged* (D and E), *positively charged* (K and R), *uncharged polar* (C, S, T, Y, N, Q, W and H) and *hydrophobic* (G, A, V, L, I, F, P and M). The categorisations of C and H are based on the similarities of their behaviour to those of the other polar residues. The amino acids may also be divided into three main groups: *charged/polar/hydrophobic*, or into just two categories like *hydrophobic/non-hydrophobic* or A/*non*-A etc, but not all of these are focused on in the present paper.

If sites are independent, the number of residues of each category (either 21 amino acids or a smaller number of categories as explained above) along a sequence will follow a multinomial distribution. The frequency of each amino acid (or other categorizations) may also be considered as a normally distributed variable.

We have for each amino acid x performed a two-way unbalanced ANOVA test for differences between M- and P-species in the amount of amino acid x. In addition to temperature, taxonomic order is included as the second factor in the ANOVA model.

The preceding compositional analysis overlooks the importance of amino acids that are gained in some contexts and lost in others. This may be a serious limitation, and one should also study the nature of the replacement patterns of residues in the aligned sequences.

2.3 Amino Acid Substitution Bias

By a comparative study of alignments it is possible to look for genome-wide amino acid directional biases in substitutions among proteins from different source populations of interest. First we examine how to model the substitutions of amino acids between two aligned sequences.

A *Substitution Pair* (SP) is defined as a pair of two amino acids, (x_M, y_P), where residue x in mesophilic group M is converted to residue y in psychrophilic group P. For a given pair of amino acids, the "forward" substitution refers to the mesophilic(M) → psychrophilic(P) direction, and the process may be called cold adaptation.

The *SP-matrix* is the count, $n_{x,y}$, of all such pairs observed in the alignment by summing over all sites where $x \neq y$. This accumulated array contains the occurrence of all position specific pairing of residues.

Furthermore, we may calculate the cumulative SP-matrices *between two* groups M and P, $N_{x,y}^{(M,P)}$, as well as *within one* group M, $N_{x,y}^{(M,M)}$, where there is no direction. The cumulative numbers are found by forming the maximum number of independent sequence pairs, were each sequence are involved only once to avoid oversampling of the data. Such possible combinations of ordered pairs of sequences in the two groups are computed, and use the average counting's in the cumulative SP-matrix.

The question we would like to address is whether there are over- or under-representations of amino acid substitutions between our temperature groups compared to a random model. We thus need to define statistical models to test for over- or under-representation in relation to relevant background distributions. A natural approach will be to consider a statistical technique for detecting adaptation by utilising the neutral expectation that aligned sequences from two species should have a symmetrical substitution matrix: for any two amino acids *x* and *y*, the number of aligned sites with *x* in one species and *y* in the second should equal the number with *y* in the first species and *x* in the second. Accordingly, in the background model we may use as our null hypothesis the 50:50 balance, and then perform a binomial test in the manner of earlier approaches [14, 15, 16].

However, we prefer a more general approach. For a given pair of amino acids, the substitution order refers to the $M \rightarrow P$ direction. The number of substitutions of residue $x_M \rightarrow y_P$ may be compared versus the following backgrounds [cf. 17]:

1. $y_P \rightarrow x_M$ [forward (*M*→*P*) vs. reverse (*P*→ *M*)]
2. $x_M \rightarrow y_M$ [forward (*M*→*P*) vs. internal (*M*→ *M*)]
3. $x_P \rightarrow y_P$ [forward (*M*→*P*) vs. internal (*P*→ *P*)]
4. $x_{M'} \rightarrow y_{P'}$ [forward (*M*→*P*) vs. forward (*M'*→ *P'*)]

Here *M'* and *P'* in case 4 are independent of the original dataset. Model 3 is usually not considered a good reference group, as the psychrophilic group is secondary with larger internal differences than the more primary mesophilic group.

Because of selective pressure or genetic drift, there may be biased substitutions on the amino acids within the mesophilic group itself. Therefore, the probability of a directional amino acid bias greater than or equal to that observed is compared relative to several relevant background frequencies (control data). Similar results for the different cases (1-4) will add weight to the conclusion.

We want to investigate whether the samples are drawn from groups with identical substitution frequencies. The null hypothesis is that the mean for the target group and the control group are the same, the alternative is that they are different. In case 1 above, only two sequences are needed as input for an independent cell counting, and the biased mutations from mesophiles to psychrophils are extracted by simply analysing the differences between the cumulative SP-matrix and its transposed matrix. In case 2 and 3 we need three sequences as input to count, and we compute two separate SP-matrices. The cumulative SP-matrices contain all the data that are used to address the question of which SPs account for the significant variations between the mesophilic and psychrophilic sequence groups. It is often convenient to look at test results from model 1 and 2 together. Model 4 may be used to find dissimilar adaptations in two different sets of alignments.

Table 1. A 2x2 table summarizing one cumulative SP-matrix with it's mesophilic background (model 2) for one particular substitution *x*→ *y*

Substitutions of *x*:	***y***	**Not *y***
Adaptation *M*→*P*:	$N_{x,y}^{(M,P)}$	$N_{x,non-y}^{(M,P)}$
Background *M*→*M*:	$N_{x,y}^{(M,M)}$	$N_{x,non-y}^{(M,M)}$

For these purposes 2x2 contingency tables were constructed, with the number of aligned sites exhibiting each of the pairwise substitution patterns. The statistical analyses are performed on the numbers from the cumulative SP-matrices, as described in Table 1.

To be able to detect substitution biases, we also *pooled* the substitution data that share the same outcome in the psychrophilic population, e.g. we studied the substitutions $x \rightarrow$ aa, where x may be any amino acid and aa is given.

Our alignment data consists of *three* different bacterial orders, and this information may be utilised to build a stratified design. In a stratified design (also called blocked analysis or matched analysis), the data are selected from two or more strata that are formed from important covariates such as taxonomic order. A separate 2x2 table is formed for each stratum. Although substitution frequencies may vary among strata, hypotheses about the overall expected frequency can be tested using the *Mantel-Haenszel* test [18, 19]. The initial data are represented as a series of K 2x2 contingency tables, where K is the number of strata. The stratification of the subjects into K disjoint groups increases the power of the test to detect association. This increase in power comes from comparing like subjects to like subjects. Simulation studies show the test to perform well in most situations with expected cell counts of at least five in most of the cells of each stratum-specific table [19].

2.4 Multiple Test Correction

Each time we statistically test the number of amino acids or SP's with a statistical test, we incur the risk of a false positive (type I error). When multiple hypothesis tests are carried out, significance levels must be adjusted to account for the increased probability of false positives. It is common practice in statistics to use a P-value threshold of 0.05 for the decision as to whether a difference is significant or not, so on average we would expect to get a false positive result about once every 20 times the test is used (1/0.05). For an experiment with 400 tests, this translates to 20 false positives (0.05 × 400 tests). This calculation ignores correlations between tests, but it is a useful guideline. It shows that a much smaller P-value threshold than 0.05 is needed to keep the number of false positives at an acceptable level.

There are many ways of correcting for this problem. The simplest is the Bonferroni method, in which each P-value is simply multiplied by the number of tests done. This method is very conservative. We can usually accept a few false positives, and a better way to establish P-value thresholds is to focus on the number of false positives that are expected at a given uncorrected P-value threshold. This approach has been formalised by Benjamini and Hochberg [20] as the false discovery rate (FDR). The idea can best be clarified by an example. If there are 20 SP's that meet a P-value threshold of 0.001, and there are 1000 SP tested, we would expect there to be 1 false positive among those 20 SP's (1000 × 0.001=1). The FDR in this case is 0.05. Benjamini and Hochberg give an easy method for finding the right P-value threshold to control the FDR at a pre-specified level [21]. We have applied the FDR analysis with a false positive level of 0.05, and results with significant difference are shown with **bold** face in the tables. This correction will in most cases lead to increased power in the statistical inference compared to the Bonferroni method.

3 Experimental Results and Discussion

3.1 Amino Acid Composition

To examine if there were detectable trends in the amino acid composition with growth temperature, amino acid frequencies of mesophilic *M* and psychrophilic *P* proteins were compared. From Figure 1 it is observed that the compositional differences between the groups are mostly marginal, with overlapping standard deviation intervals. Despite this, there seems to be differences in composition of amino acids A, G and Y that possibly may be symptomatic for cold adaptation of this protein.

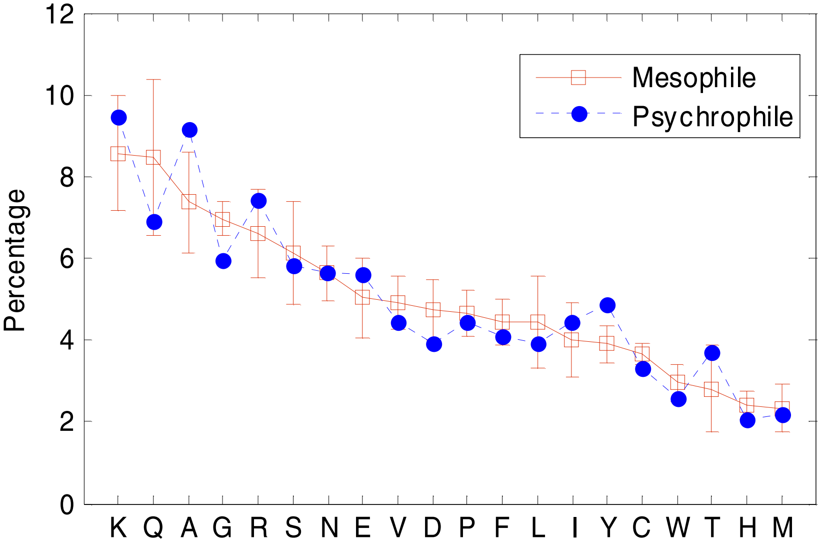


Fig. 1. Comparison of amino acid compositions of the two temperature groups with the amino acids ranked according to frequencies in the mesophile group. Error bars represent the empirical standard deviations of the mesophile sequences, while the overlapping error bars from the psychrophile sequences are not shown.

The temperature of the environment of the organism seems to be important for which amino acid is favorable in its proteins, and the frequency shifts depending on temperature appears to be a trend independent of lineage. For each amino acid we calculate a P-value for rejecting the null hypothesis that the groups did not vary, and perform the False Discovery Rate (FDR) multiple testing correction [20], setting the false discovery rate to be 5%. The two-way statistical analyses, with temperature and taxonomy as factors, show in table 2 that the decrease of amino acid G, and the increase of A and Y, can be significantly related to temperature. Changes of A and G are mainly located at the protein surface, Y at the interior, and G also in loop regions (data not shown). The high mutability of A is probably due to its role as a default residue. Lack of gamma-carbon also allows substitutions with small steric obstructions, and replacements to the slightly rigid A induce smaller changes in the fold than substitutions to the very flexible G.

It is interesting to observe that the compositional change of the amino acids A and G with T_{opt} are in agreement with earlier results found by Nakashima et al. [22], mainly from a study of proteins from a higher temperature range, while Y is opposite.

Counts of H, M, and W in the present study are low (< 5), making the statistical test questionable, and to increase counts we joined data in broader categories, as also shown in table 2. Divergences related to taxonomic order are shown in the table, but not discussed further in this article.

Table 2. P-values from two-way ANOVA tests of effect of temperature and taxonomic order on amino acid composition. Differences with a false positive level <0.05 are shown in **bold** (the FDR-corrected threshold is 0.012 in upper table, and 0.024 in lower).

aa	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V
P_{tmp}	**.0003**	.02	.80	.08	.09	.08	.06	**.003**	-	.16	.09	.09	-	.26	.63	.76	.06	-	**.001**	.24
P_{tax}	**.004**	**$<10^{-4}$**	.79	.24	.06	**.01**	.27	.80	-	**.0004**	**$<10^{-4}$**	.88	-	.37	.52	.94	.67	-	**.005**	.04

aa group	Hydrophobic	Charged	Negative	Positive	Polar
P_{tmp}	.92	**.02**	.95	**.004**	.86
P_{tax}	**.01**	**.003**	.11	**.002**	.69

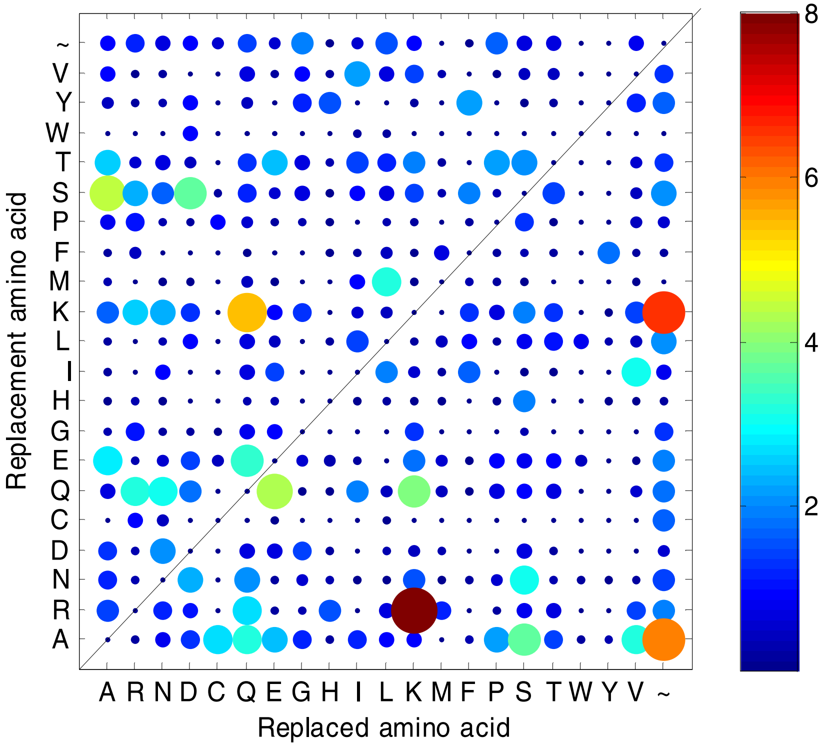


Fig. 2. Visualisation of the mean numbers of individual substitutions, observed from the mesophilic to the psycrophilic group, summed over all the strata. K, R, A and Q are clearly the most central amino acids in terms of involvement. A tilde (~) indicates a gap.

Table 3. P-values from pooled substitutions, where the amino acids are the outcome in the psychrophilic group. P-values are obtained from stratified Mantel-Haenszel test. Model 2 only contains two strata, because the third strata only included one mesophilic sequence. The tilde (~) indicates gap. Differences with a false positive level <0.05 are shown in **bold** (the FDR-corrected threshold is 0.005 in upper table, and <0.02 in lower).

Subst to	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	~
Model 1	**.005**	.08	.92	.11	.28	.07	.22	.03	.26	.41	.31	.12	.94	.33	.63	.91	.04	.19	**.001**	.47	**.003**
Model 2	.24	.26	.86	.92	**.001**	.64	.67	.07	.36	.51	.99	.01	.27	.15	.82	.99	.10	-	.17	.84	.008

aa group	Hydrophobic	Charged	Negative	Positive	Polar
Model 1	.79	.13	.87	.02	.93
Model 2	.29	.90	.65	.99	.19

3.2 Analysis of Substitution Patterns

To better understand the individual contributions by substitutions, Figure 2 reports counts of all the 380 substitution pairs (421 when gaps are included) added over the three strata. Because of low counts, the data may not be analysed by the full 21x21 substitution matrices, and the number of categories were reduced to 2x2 representation, as described in Table 1. The data were also analysed against different backgrounds, as defined in Materials and Methods above. The pooled cumulative SPs were all tested for statistical significance by using the two-sided Mantel-Haenszel test. The tests were performed with an assumption that the significant changes seen in substitution patterns may be interpreted as due to thermal adaptation. A summary of the results with the substitution biases in the ($M \rightarrow P$) direction are shown in Table 3.

The Mantel-Haenszel tests based on substitutions are confirming the ANOVA test results based on compositions.

In the psychrophilic enzyme we find no significant differences in the number of hydrophobic or polar residues, compared with the mesophilic enzyme. But the exterior of the psychrophilic enzyme has a more positive electrostatic charge. A more positive (or negative) charged surface is considered beneficial for cold adaptation, because it will increase the solubility of the protein in water. Another possible effect is that the extra positive amino acids may repel each other and thereby increase the flexibility of the molecule.

The statistical approach above, assumes pairwise independence also among the sequences samples (and species) in each temperature group at each strata. This is not the case in the presence of underlying phylogenetic processes with horizontal gene transfer and recombination. Hence, we may modify the approach to treat each temperature group at each stratum as one observation. When there are multiple dependent sequence samples in a group, instead of computing the cumulative SP-matrix, we compute the average counting's $\overline{n}_{x,y}$ in the representative SP-matrix:

$$\overline{n}_{x,y}^{(M,P)} = \tfrac{1}{G_{MP}} \sum n_{x,y}, \qquad x \neq y$$

where G_{MP} is the number of ordered pairs of sequences in the two groups. The use of group sample mean also removes some of the random genetic drift ("noise"). This way of arriving at mean cell counts may also be based on a rationale of assigning a weight to every sequence as the inverse of the number of dependent sequences in its group, where all sequences are given equal weights that sum to 1. Then we just apply a counting technique by counting the multiplied weights of the substitution pairs between the sequences in the two groups.

In the same way we may treat the data from the compositional analyses by applying average counts of each amino acid within group *M* and *P* at each stratum.

We computed P-values based the modified models, and arrived at the following P-values for the most significant results found in Table 1 and 2: A: 0.10 and 0.07, G: 0.06 and 0.23, Y: 0.07 and 0.03, charged: 0.57 and 0.66, positive: 0.006 and 0.15. The P-values are from the temperature factor of the ANOVA test, and from model 1 in the Mantel-Haenzel test, respectively. The modified models and tests are more conservative, but show some of the same results as previously found, although the results in general are weaker.

4 Conclusion

As a general observation, it seems difficult to resolve fixed and absolute elements of cold adaptation of a particular protein molecule at the overall compositional level, or based on the general substitution matrix. However, a *stratified* two-way design, where also taxonomic order is taken into consideration, yields a better statistical analysis. This model is more robust against differences due to differences between different phylogenetic lineages, and hence appropriate to detect possible temperature associated trends independent of linage. On our dataset with three strata, both ANOVA test based on composition and the Mantel-Haenszel test based on substitution patterns performed well. The two tests show some of the same trends, but with more strata the Mantel-Haenszel test may be more powerful, although the missing categorisation of gaps represents a problem in the substitution analyses.

Both the compositional and the substitution statistical analyses reveal that psychrophilic and mesophilic proteins have similar hydrophobic contributions. However, the exterior of the psychrophilic molecule is found to be significantly more positively charged. A positively charged molecule will increase its solubility in water.

Several minor structural and charge increasing substitutions appear to be responsible for the cold adaptation, rather than a particular substitution. There must also be some fine-tuning in the process, such as increasing charged residues mainly at the exterior of the protein [22]. It is also possible that there are several ways to adapt within each enzyme class. The endonuclease studied here binds the negatively charged phosphate backbone of DNA [7]. It is therefore interesting to notice an increase in positively charged amino acids, where Lys (K) and Arg (R) are favoured amongst the cold adapted sequences. Our compositional analyses also imply a somewhat reduced content of negative charged Asp (D). The higher positive electrostatic potential will increase the electrostatic interactions between the enzyme and the negatively charged substrate, and may in this way contribute to an increase in catalytic efficiency.

Acknowledgements

The sequence alignment of *endonuclease I* was kindly provided by Bjørn Altermark.

Our algorithms and data can be downloaded as part of our Matlab toolbox *DeltaProt* at: http://www.math.uit.no/bi/deltaprot/

References

1. Committee on Frontiers in Polar Biology (CB). Frontiers in polar biology in the genomics era. Washington, DC, USA: National Academies Press, 2003.
2. Sohail, K. and Cavicchioli, R.: Cold-adapted enzymes. Annu. Rev. Biochem. 75: 403-433, 2006.
3. Saunders, N.F.W., Thomas, T., Curmi, P.M.G., et al.: Mechanisms of thermal adaptation revealed from the genomes of the Antarctic Archaea Methanogenium frigidum and Methanococcoides burtonii. Genome Res 13 (7): 1580-1588, 2003.
4. Karlin, S., Brocchieri, L., Trent, J., Blaisdell, B.E., Mrazek, J.: Heterogeneity of genome and proteome content in bacteria, archaea, and eukaryotes. Theor Popul Biol. 61:367–390, 2002
5. Pe'er, I., Felder, C.E., Man, O., Silman, I., Sussman, J.L., Beckmann, J.S.: Proteomic signatures: Amino acid and oligopeptide compositions differentiate among phyla. Proteins-Structure Function and Genetics 54 (1): 20-40, 2004.
6. Jekel, M. and Wackernagel, W. The periplasmic endonuclease I of Escherichia coli has amino-acid sequence homology to the extracellular DNases of Vibrio cholerae and Aeromonas hydrophila. Gene 154(1): 55-9, 1995.
7. Li, C. L., Hor, L. I., Chang, Z. F., Tsai, L. C., Yang, W. Z. and Yuan, H. S. DNA binding and cleavage by the periplasmic nuclease Vvn: a novel structure with a known active site. Embo J 22(15): 4014-25, 2003.
8. Hall, T.A.: BioEdit: a user-friendly biological sequence alignment editor and analysis program for Windows 95/98/NT. Nucl. Acids. Symp. Ser. 41: 95-98, 1999.
9. Lambros, R.J., Mortimer, J.R. and Forsdyke, D.R.: Optimum growth temperature and the base composition of open reading frames in prokaryotes. Extremophiles 7: 443-450, 2003.
10. Bendtsen, J.D., Nielsen, H., von Heijne, G. and Brunak, S. Improved prediction of signal peptides: SignalP 3.0. J. Mol. Biol. 340: 783-795, 2004.
11. Huang, S.L. et al. PGTdb: a database providing growth temperatures of prokaryotes. Bioinformatics, 20: 276-278, 2004.
12. Garrity, G.M.: Bergey's Manual of Systematic Bacteriology, Vol. 2B. Plenum US, 2nd edition, 2005.
13. Fraczkiewicz, R. and Braun, W.: Exact and efficient analytical calculation of the accessible surface areas and their gradients for macromolecules. J. Comp. Chem. 19(3): 319-333, 1998.
14. Haney, P. J., Badger, J. H., Buldak, G. L. et al.: Thermal adaptation analyzed by comparison of protein sequences from mesophilic and extremely thermophilic Methanococcus species. PNAS vol. 96 (7) 3578-3583, 1999.
15. McDonald, J.H., Grasso, A.M., Rejto, L.K.: Patterns of temperature adaptation in proteins from Methanococcus and Bacillus. Molecular Biology and Evolution 16 (12): 1785-1790, 1999.
16. Smith, N.G.C., Eyre-Walker, A.: A test of amino acid reversibility. J Mol Evol 52: 467-469, 2001.

17. Chakravarty, S. and Varadarajan, R.: Elucidation of factors responsible for enhanced thermal stability of proteins: A structural genomics based study. Biochemistry 41 (25): 8152-8161, 2002.
18. Mantel, N. and Fliss, J.L.: Minimum expected cell-size requirements for the Mantel-Haenszel one-degree-of-freedom chi-square test and a related rapid procedure. American Journal of Epidemiology 112: 129-134, 1980.
19. Parshall, C.G., Miller, T.R.: Exact versus asymptotic Mantel-Haenszel DIF statistics - A comparison of performance under small-sample conditions. Journal of Educational Measurement 32 (3): 302-316, 1995.
20. Benjamini, Y. and Yekutieli, D.: The control of the false discovery rate in multiple testing under dependency. Ann Stat 29 (4): 1165-1188, 2001.
21. Koen, J.F. et al.: Implementing false discovery rate control: increasing your power. OIKOS 108: 643-647, 2005.
22. Nakashima, H., Fukuchi, S., Nishikawa, K.: Compositional changes in RNA, DNA and proteins for bacterial adaptation to higher and lower temperatures. Journal of Biochemistry 133 (4): 507-513, 2003.

Markov Model Variants for Appraisal of Coding Potential in Plant DNA

Michael E. Sparks[1], Volker Brendel[1,2], and Karin S. Dorman[1,2]

[1] Department of Genetics, Development and Cell Biology, Iowa State University, Ames, IA 50011, USA
[2] Department of Statistics, Iowa State University, Ames, IA 50011, USA

Abstract. Markov chain models are commonly used for content-based appraisal of coding potential in genomic DNA. The ability of these models to distinguish coding from non-coding sequences depends on the method of parameter estimation, the validity of the estimated parameters for the species of interest, and the extent to which oligomer usage characterizes coding potential. We assessed performances of Markov chain models in two model plant species, *Arabidopsis* and rice, comparing canonical fixed-order, χ^2-interpolated, and top-down and bottom-up deleted interpolated Markov models. All methods achieved comparable identification accuracies, with differences usually within statistical error. Because classification performance is related to G+C composition, we also considered a strategy where training and test data are first partitioned by G+C content. All methods demonstrated considerable gains in accuracy under this approach, especially in rice. The methods studied were implemented in the C programming language and organized into a library, `IMMpractical`, distributed under the GNU LGPL.

1 Introduction

Markov chain models, as applied to problems concerning gene recognition in DNA sequences, make the fundamental assumption that sequences of different functional roles exhibit distinct and reproducible dependencies among adjacent nucleotides, such that sequences can be distinguished by oligomer usage. In practice, Markov models appear to be a suitable proxy to the (unknown) generative models that have produced biologically relevant nucleic acid sequences, and they have enjoyed widespread use in popular gene prediction applications, including `GENSCAN` [1], `GlimmerM` [2], and `GeneMark.HMM` [3]. The Markov models used in these applications tend to be complex, and in most cases, only heurisitic procedures exist for estimating Markov transition probabilities. Because both the validity of the Markov model assumption and the accuracy of the estimation procedures are unknown, it remains important to assess classification performance in novel applications. As this study is primarily motivated by the need to annotate plant gene structures, we used sequences from the model plant species *Arabidopsis thaliana* and *Oryza sativa* (rice).

There are a number of distinct methods for estimating Markov chain transition probabilities and selecting among models of varying complexity. Azad and

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 394–405, 2007.

Borodovsky [4] undertook an empirical survey of fixed-order, χ^2-interpolated [5,6], and top-down deleted interpolated Markov models [7] in prokaryotic taxa, and found considerable differences in the relative performances of each method as a function of genomic sequence characteristics, particularly G+C composition. The present study extends this work by comparing a greater breadth of training methods and by considering method performances in the context of two eukaryotic taxa. We show that, for the task of binary classification of coding and intron sequences from *A.thaliana* and rice, all the Markov model variants surveyed here (canonical fixed-order, χ^2-interpolated, top-down and bottom-up deleted interpolated [7]) performed approximately equally. All Markov model variants were implemented in the C programming language and organized into a library, called `IMMpractical`, which is distributed under the GNU lesser/library general public license and is available for download at [8].

We also compared a *standard approach* that trains and tests without concern for the G+C composition of sequences, and a *quartiled approach* in which sequences are first partitioned into quartiles on the basis of overall G+C content, and quartile-specific transition probability estimates are used to classify. The latter strategy resulted in substantial improvements in classification accuracy relative to the standard approach, particularly in rice.

2 Materials and Methods

2.1 Data Accumulation

The success of any gene-finding algorithm to accurately classify sequences is largely dependent on how well the training data represent true coding and non-coding DNA. To obtain a reliable set of nuclear protein coding and intron sequences for training and testing purposes, we started with the current genome annotations for *A.thaliana* and rice available from the TAIR (version 6.0, [9]) and TIGR (version 4.0, [10]) resources, respectively.

As we were primarily interested in distinguishing coding sequences from introns in split genes, single exon genes were excluded. This exclusion also eliminated many processed pseudogenes, which are often intronless and share similar features with functional genes [11,12]. We ignored all loci with multiple gene models because these may be alternatively spliced [13], making coding/intron classification much more difficult [14].

Full-length coding sequences were parsed from assembled pseudomolecules based on reference coordinates, and if any ambiguous nucleotide symbols were encountered, the gene was discarded. Start and stop codons along with 5'- and 3'-UTRs were removed from the coding sequences, and only genes encoding translation products of 150 or more amino acids were retained. The selected sequences were compared to the TIGR plant repetitive element database [10] using `BLASTN` [15], and all coding sequences with significant matches (E-value $< 10^{-15}$) were removed. We also used BLAST to limit redundancy in the coding data by randomly retaining only one member of each pair of sequences having at least

80% nucleotide identity over at least 80% of the length of both sequences. Reduction in dataset size during this refinement process is indicated in Table 1. Introns from the remaining gene structures were parsed from the pseudomolecules, leaving the concatenated exons as the coding data set. Introns that exceeded 50 nucleotides in length and contained no ambiguous characters were retained, and the resulting collection was made non-redundant using BLAST, as described above, to form the intron dataset. In total, we retained 15,538 coding sequences (mean length 1,467nt) and 87,477 intron sequences (mean length 159nt) from *A.thaliana*. In rice, 24,349 coding sequences (mean length 1,502nt) and 104,737 intron sequences (mean length 396nt) were retained. For the quartiled approach, coding and intron sequences were separated into quartiles according to their overall percent G+C composition (see Fig. 1).

Table 1. Number of genes excluded in the *A.thaliana* and *O.sativa* data sets at each refinement stage

Type Removed	*A.thaliana*	*O.sativa*
Annotated pseudogenes	3,818	0
Intronless genes	5,793	12,780
Alternatively spliced genes	2,887	4,280
Genes with ambiguous nucleotides	4	20
Genes with protein length $<$ 150	2,009	7,433
Repetitive elements	60	7,379
Redundant genes	250	322
Total remaining	15,538	24,349

2.2 Fixed-Order Markov Models (FO)

For fixed-order methods, an order, k, is selected for the Markov chain based on empirical or statistical considerations—in practice, this is often set to five [1,3,11], and we also used this value. Let h_k represent some pretext, or history, of length k that precedes a nucleotide i. The fixed-order Markov chain has 4^{k+1} transition probabilities, whose maximum likelihood estimates are

$$\widehat{P}(i \mid h_k) = \frac{\mathrm{Cnt}(h_k, i)}{\sum_{j \in \{A,C,G,T\}} \mathrm{Cnt}(h_k, j)}, \tag{1}$$

where $\mathrm{Cnt}(h_k, x)$ is the count of oligomer h_k succeeded by some nucleotide x in the training data. Given a test sequence $S = s_1 s_2 \cdots s_n$ of length n, the likelihood, assuming that the sequence belongs to some functional class t with maximum likelihood estimates $\widehat{\Omega}_t = \{\widehat{P}(i \mid h_k)\}$, is

$$P(S \mid t, \widehat{\Omega}_t) = \prod_{j=1}^{n-k} \widehat{P}\left(s_{j+k} \mid s_j s_{j+1} \cdots s_{j+k-1}\right).$$

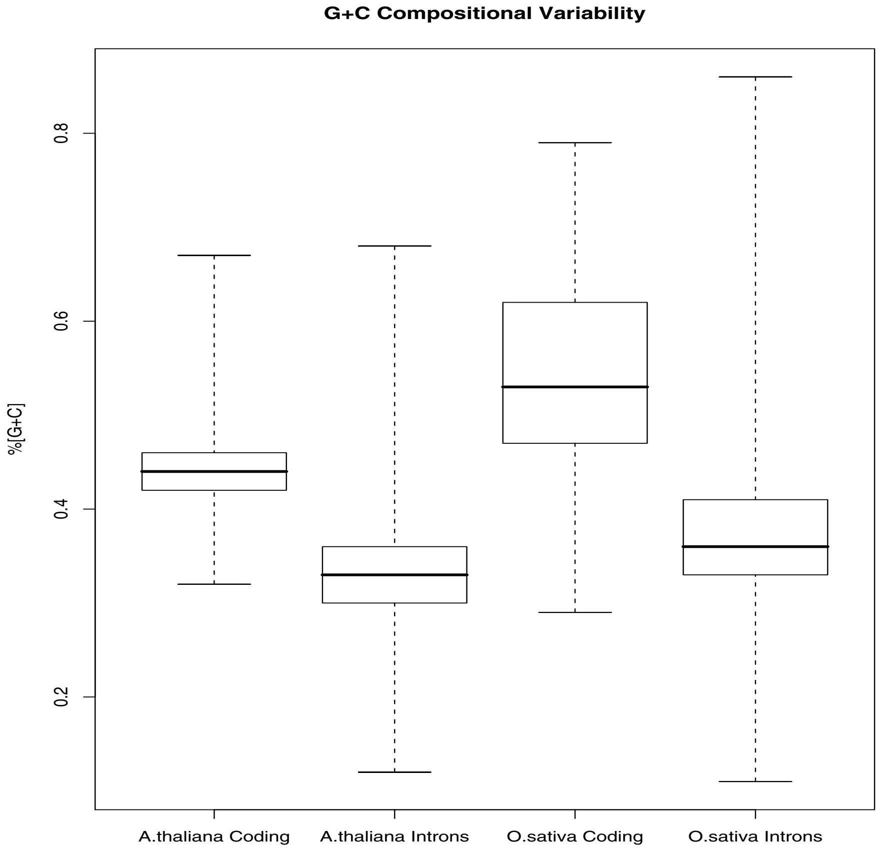


Fig. 1. Box-and-whiskers plot showing variation in G+C percent composition for coding and intron sequences in *A.thaliana* and *O.sativa*. Each data set was partitioned on the quartiles, such that partitions contained roughly 3,884 coding and 21,869 intron sequences for *A.thaliana* and roughly 6,087 coding and 26,184 intron sequences for *O.sativa*, respectively.

For coding sequences, one recognizes the distinct properties of the three codon positions by computing one set of transition probabilities $P^{(f)}(i \mid h_k)$ for each of the three reading frames, $f = 1, 2, 3$. There are now $3 \times 4^{k+1}$ parameters to estimate for this inhomogeneous Markov chain model, and each is estimated by Eq. (1) with oligomer counts from the appropriate codon position. When simultaneously modeling a coding sequence shadow, parameters are also estimated for the three codon positions in the reverse complement [16].

Certain rare oligomers may not appear in the training data, resulting in null transition probabilities using Eq. (1). Any test sequence containing an unobserved oligomer is then impossible (has zero likelihood) under the estimated model. To avoid this problem, we use parameter smoothing, where for all

h_k and i, $\mathrm{Cnt}(h_k, i)$ is incremented by a fixed integer (in practice, five), ensuring at least a basal representation of all possible oligomers in the training data.

2.3 Interpolated Markov Models (IMMs)

The general paradigm of IMMs is that each transition probability is determined by taking linear, weighted sums of relevant fixed-order transition probabilities. For the transition probability with context h_k, fixed order transition probabilities for the pretext of length k and all shorter pretexts are used to produce the smoothed transition probability

$$P_{\mathrm{imm}}(i \mid h_k) = \sum_{x=-1}^{k} \mu_x(h_k)\widehat{P}(i \mid h_x).$$

Here, $\widehat{P}(i \mid h_{-1})$ is taken to be one over the cardinality of the nucleotide alphabet, i.e., 0.25. $P_{\mathrm{imm}}(i \mid h_k)$ is a probability when the weights $\mu_x(h_k)$ satisfy $0 \leq \mu_x(h_k) \leq 1$ for all x and $\sum_{x=-1}^{k} \mu_x(h_k) = 1$. To account for data sparsity, these models assign weights in terms of oligomer frequencies, preferentially giving more weight to oligomers with longer histories, unless they occur rarely enough in training data that more weight should be given to one of their 5'-truncated variants. Final, smoothed transition probabilities of oligomers whose histories do not occur in the training data are defined as $P_{\mathrm{imm}}(i \mid h_k) = P_{\mathrm{imm}}(i \mid h_z)$, where $z = \max z' \in [1, k) : \mathrm{Cnt}(h_{z'}) > 0$ and k is the maximum Markov chain order. We consider three distinct methods for estimating the smoothed transition probabilities as described in [4,5,6,7].

χ^2-Interpolated Markov Models (χ^2). The χ^2-IMM defines transition probabilities iteratively as

$$P_{\mathrm{chi}}(i \mid h_k) = \lambda(h_k)\widehat{P}(i \mid h_k) + [1 - \lambda(h_k)]P_{\mathrm{chi}}(i \mid h_{k-1}), \tag{2}$$

with boundary condition $P_{\mathrm{chi}}(i \mid h_{-1}) = \widehat{P}(i \mid h_{-1})$. The history weights for $x = 0, \ldots, k$ are

$$\lambda(h_x) = \begin{cases} 1 & \text{if } \mathrm{Cnt}(h_x) \geq T; \\ 0 & \text{if } \mathrm{Cnt}(h_x) < T \text{ and } q < 0.5; \\ \frac{q \times \mathrm{Cnt}(h_x)}{T} & \text{otherwise.} \end{cases}$$

T is some minimally-reliant count threshold for pretexts, e.g., 400; and q is the confidence (one minus the p-value) that the distribution of $i \mid h_x$ differs from that of $i \mid h_{x-1}$, $i \in \{A, C, G, T\}$, obtained by a χ^2 statistical test [5,6].

One possible scenario that is not addressed in any literature we encountered describing χ^2-IMMs [5,6,17,4] is the condition where some pretext h_y occurs more than T times in the training data, but $i \mid h_y$ does not occur for some nucleotide i. Then recursion (2) for computing $P_{\mathrm{chi}}(i \mid h_x)$ can generate problematic null transition probabilities, precisely the complication interpolated models were developed to avoid. For such cases, we used an approach similar to that described in [18] for correcting weight array matrices in splice site modeling:

$$P_{\text{fix}} = \frac{1}{\text{Cnt}(h_y)}$$
$$P_{\text{new}} = P_{\text{old}}(1 - 4 \times P_{\text{fix}}) + P_{\text{fix}},$$

where any null transition probability is re-assigned the value P_{fix}, and all remaining non-null probabilities in the distribution are adjusted to P_{new} as a function of P_{fix} and their previous values, P_{old}. Alternative solutions are described in [7].

Top-down Deleted IMMs (TDDI). The basic idea of deleted IMMs is to divide the training data into a large *development* set (D) and a small *heldout* set (H)—the development set generates initial, unrefined transition probability estimates according to Eq. (1), which are generalized to the heldout set by cross-set maximization [7]. To prevent over-fitting to the heldout set, pretexts in the development set are partitioned into groups based on their frequencies, and all pretexts in a group are tied to the same weight. The pretext partitions are called buckets $B_{x,m} = \{ h_x : \text{bound}_{x,m-1} \leq \text{Cnt}_D(h_x) < \text{bound}_{x,m}\}$, where x indexes pretext length, m indexes the bucket, and Cnt_D indicates counts in D only. Bucket width is specified using a real-valued constant (e.g., 1.2, which is used in our implementation) dictating ratios of adjacent bucket boundaries.

Top-down deleted IMM-smoothed probabilities are computed by recursively solving, for $x = 0, 1, \ldots, k$,

$$P_{\text{TD}}(i \mid h_x) = \lambda_m(h_x)\widehat{P}(i \mid h_x) + [1 - \lambda_m(h_x)]P_{\text{TD}}(i \mid h_{x-1}), \tag{3}$$

with $\lambda_m(h_x)$ values computed as

$$\underset{0<\lambda<1}{\text{argmax}} \left\{ \sum_{\substack{i\in\{A,C,G,T\} \\ h_x \in B_{x,m}}} \text{Cnt}_H(h_x, i) \log \left[\lambda\widehat{P}(i \mid h_x) + (1 - \lambda)P_{\text{TD}}(i \mid h_{x-1}) \right] \right\}, \tag{4}$$

and $P_{\text{TD}}(i \mid h_{-1}) = \widehat{P}(i \mid h_{-1})$ again initializes the recursion.

Bottom-up Deleted IMMs (BUDI). In the bottom-up deleted IMM approach, development pretexts of length k are partitioned into buckets $B_{k,m}$ in similar fashion to the top-down variant. Each BUDI transition probability $P_{\text{BU}}(i \mid h_k)$ is produced through a series of iterations initialized with

$$P^{(k)}(i \mid h_k) = \xi\widehat{P}(i \mid h_{-1}) + (1 - \xi)\widehat{P}(i \mid h_k), \tag{5}$$

for $\xi = 10^{-5}$. The recursion formula for the smoothing procedure is

$$P^{(l-1)}(i \mid h_k) = \lambda_{l,m}(h_k)P^{(l)}(i \mid h_k) + [1 - \lambda_{l,m}(h_k)]\widehat{P}(i \mid h_{l-1}), \tag{6}$$

starting at $l = k$ and producing $P_{\text{BU}}(i \mid h_k) := P^{(-1)}(i \mid h_k)$ upon termination when $l = 0$. Weighting factors $\lambda_{l,m}(h_k)$ for the recursion are computed as

$$\underset{0<\lambda<1}{\text{argmax}} \left\{ \sum_{\substack{i\in\{A,C,G,T\} \\ h_k \in B_{k,m}}} \text{Cnt}_H(h_k, i) \log \left[\lambda P^{(l)}(i \mid h_k) + (1 - \lambda)\widehat{P}(i \mid h_{l-1}) \right] \right\}. \tag{7}$$

2.4 Accounting for G+C Content

We compared two approaches for fitting and using Markov chains with our data sets. The default method—the *standard approach*—involved producing a single set of transition probability estimates by training with all available data from each cross validation replicate; the same estimates were used to assay all test fragments. We also considered a *quartiled approach*, in which all sequences available for a given species were classified into quartiles on the basis of overall G+C composition (See Fig. 1). Coding and intron training sequences were quartiled separately, and quartile-specific Markov chains were estimated. Each test sequence was assigned a quartile based on its G+C content and likelihoods were computed using the appropriate Markov chains.

2.5 Test Design

The estimation methods were assessed on their abilities to correctly identify the—*a priori* known—functional class of a test sequence using the familiar Genmark framework [16]. Only binary classification of sequences as either coding or intron was tested. Likelihoods of test data were computed under $N = 7$ Markov models: six coding Markov models for each frame of the forward f_1, f_2, f_3 or shadow w_1, w_2, w_3 strand; and a homogeneous Markov chain itr for the intron hypothesis. Prior probabilities are specified as follows. Let z be the (hypothesized) sequence type; then the prior is

$$P(z) = \begin{cases} 1/2 & \text{if } z = itr; \\ \frac{1}{(N-1)\times 2} & \text{otherwise.} \end{cases} \tag{8}$$

Bayes rule provides the classifier:

$$P(z \mid S) = \frac{P(S \mid z) \times P(z)}{P(S)}, \tag{9}$$

where S is a test sequence. $P(\text{intron} \mid S) = P(itr \mid S)$ is obtained directly from Bayes rule, and $P(\text{coding} \mid S) = \sum_{z\in\{f,w\}} \sum_{i=1}^{3} P(z_i \mid S)$. A sequence was classified as coding if $P(\text{coding} \mid S)$ exceeded 0.5—otherwise it was labeled as an intron.

We used a five-fold cross validation approach where, for each cyclic permutation, transition probabilities for each of the coding and intron classes were estimated using four of the data partitions, and methods were assayed against the remaining test partition. (Note that for the deleted IMM variants, three of the five data partitions were used for the development set, and one for the held-out.) Results from all five cross-validation replicates were pooled and averaged for final reporting.

To establish uniformity in training and testing sample sizes, we reduced the sizes of the five initial data partitions by randomly sampling a subset from each partition. For each species and sequence type, 3,000 random sequences were retained for the standard approach, and 750 from each bin in the quartiled data. For test samples used under the standard approach, 2,500 fragments were randomly sampled from each test partition, for each of the coding and intron data sets, independently for both species; similarly, we randomly sampled 750 fragments from each bin in the quartiled method.

Normalizing for test sequence length is crucial for comparing performances of the methods at classifying sequences—longer test sequences would increase the odds of detecting a signal characteristic of the underlying generative model, and would tend to increase classification accuracy relative to shorter fragments. A fixed length of 96 nucleotides was used for assaying the methods under both the standard and quartiled approaches. A single test fragment was randomly parsed from each sequence among the test data partitions.

3 Results

Table 2 presents the average classification success of the Markov model training variants under the standard approach, for both species. Although the χ^2-IMM achieved the maximum accuracy in all but one category, this advantage was not statistically significant. The only statistically significant differences (p-values < 0.01) were the poorer performance of FO compared to all three IMM variants in *A.thaliana* and the poorer performance of BUDI relative to the other IMM variants in rice. Notably, all methods were significantly less successful at the classification task in rice relative to *A.thaliana*.

Table 2. Mean success rates, averaged over five cross-validation replicates, for *A.thaliana* and *O.sativa* coding and intron sequences, under the standard approach. Values are given as percentages and standard deviations are shown in parentheses.

	A.thaliana			*O.sativa*			Overall
	Coding	Intron	Averaged	Coding	Intron	Averaged	
FO	96.78 (0.16)	94.96 (0.29)	95.87 (0.22)	87.15 (1.12)	86.89 (0.90)	87.02 (1.01)	91.44 (0.62)
TDDI	97.16 (0.24)	95.28 (0.45)	96.22 (0.34)	87.20 (0.61)	87.30 (0.65)	87.25 (0.63)	91.73 (0.49)
BUDI	96.95 (0.34)	94.99 (0.63)	95.97 (0.48)	86.28 (0.93)	86.45 (1.09)	86.37 (1.01)	91.17 (0.75)
χ^2	97.20 (0.22)	95.31 (0.41)	96.25 (0.32)	87.42 (0.59)	87.29 (0.74)	87.36 (0.67)	91.81 (0.49)

We noticed that the success of classification varied considerably depending on the G+C content of the test sequence (data not shown). To address this problem, we partitioned the data into quartiles based on G+C content and trained quartile-specific Markov chains (the quartiled approach). Under this approach, classification performance still depended on G+C content as shown in Table 3 (only χ^2-based results are shown, though all methods exhibit similar patterns). For coding sequences, method performance generally increased slightly with G+C composition, except in the fourth quartile, where a slight tapering in prediction accuracy was seen. In contrast, performance generally decreased as G+C composition increased for intron sequences, with a marked drop in performance in the fourth quartile.

Table 3. Quartile-specific mean coding and intron fragment identification success rates for the χ^2-interpolated method, averaged over all five cross-validation replicates. Values are given as percentages and standard deviations are shown in parentheses.

Species	Class	1st	2nd	3rd	4th
A.thaliana	coding	98.11 (0.71)	99.47 (0.25)	99.41 (0.18)	99.12 (0.28)
	intron	99.89 (0.11)	99.55 (0.20)	98.61 (0.20)	92.85 (0.66)
O.sativa	coding	94.93 (0.88)	97.84 (0.53)	99.44 (0.15)	98.46 (0.58)
	intron	99.71 (0.15)	99.47 (0.14)	99.33 (0.16)	88.69 (1.65)

Despite the continued G+C content-dependent performance differences, the quartiled approach achieved a clear performance gain over the standard approach for all training methods (Table 4). The performance boost was moderate—though significant—for *A.thaliana* (roughly $2 - 3\%$) and even more dramatic for rice (roughly 10%). Importantly, all measures of classification performance improved under the quartiled approach.

Table 4. Comparison of classifiers under standard (std) and quartiled (qrt) approaches. Predictions across cross-validation replicates were pooled for a total of 30,000 distinct test cases. Classification measures, per [19], are Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$; Sn (Sensitivity) = $\frac{TP}{TP+FN}$; Sp (Specificity) = $\frac{TP}{TP+FP}$; Corr. Co. (Correlation Coefficient) = $\frac{TP \times TN - FP \times FN}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}}$; ROC AUC (Area under receiver operator characteristic curve, calculated using [20]); where TP are true positives; FP, false positives; TN, true negatives; FN, false negatives.

		FO		TDDI		BUDI		χ^2	
		std	qrt	std	qrt	std	qrt	std	qrt
A.thaliana	Accuracy (%)	95.83	98.01	96.29	98.41	96.11	98.42	96.27	98.38
	Sn (%)	96.84	98.53	97.29	99.04	97.21	98.99	97.33	99.03
	Sp (%)	94.93	97.52	95.39	97.81	95.12	97.88	95.31	97.76
	Corr. Co.	0.92	0.96	0.93	0.97	0.92	0.97	0.93	0.97
	ROC AUC (%)	99.02	99.67	99.11	99.73	99.08	99.74	99.11	99.73
O.sativa	Accuracy (%)	86.84	96.89	87.11	97.22	86.37	97.26	87.25	97.24
	Sn (%)	86.80	97.35	86.96	97.61	86.01	97.77	87.17	97.67
	Sp (%)	86.87	96.45	87.23	96.86	86.63	96.77	87.31	96.83
	Corr. Co.	0.74	0.94	0.74	0.94	0.73	0.95	0.75	0.94
	ROC AUC (%)	93.91	99.03	94.06	99.12	93.52	99.08	94.11	99.14

4 Discussion

While the gene structure prediction community has increasingly turned to gene annotation approaches dependent on homology information [21,22,23], the continued development of single-genome *ab initio* gene prediction tools remains worthwhile. Multi-genome gene prediction requires the presence of syntenic regions from two or more moderately divergent genomes. Genomic sequences from related taxa do not always exist, and the optimal level of evolutionary divergence between such genomes remains unknown [22]. Indeed, even if requisite genomic data were abundant for all such gene annotation tasks, and the models worked perfectly, these methods would restrict attention to shared, homologous gene structures. Arguably, the complement of unique, species-specific genes, e.g., novel antifreeze glycoproteins in Arctic fish [24] and sex pheromones in moths [25], would be considerably more interesting for further experimental characterization by biologists. Thus, demand for highly sensitive single-genome gene prediction methods persists.

We have assayed the relative performances of a number of transition probability estimation methods for Markov chain models on coding and intron sequences of varying G+C composition in the model plant species *Arabidopsis thaliana* and *Oryza sativa*. Computational gene finders produced most gene annotations used to form our dataset [9,10], which could have biased the data to favor one model over another. Because prediction methods are not recorded [13], we were unable to test or correct for such bias. However, Fisher's exact tests on the accuracies we have computed show that most methods perform equivalently in the standard approach. Only FO is significantly worse in *A.thaliana*, and BUDI is significantly worse than the other IMM variants in rice. The fixed order model becomes statistically less accurate in both plant species under the quartiled approach, but the order $k = 5$ may not be appropriate for the reduced size of quartiled data sets.

It is well known that classification success depends on G+C content (e.g., [1,4]). We observed that misclassified coding fragments are generally G+C-poorer than usual, while misclassified intron fragments are generally G+C-richer. In fact, the G+C profile of misclassified fragments loosely mimics that of correctly classified fragments in the competing functional category (data not shown). Markov models perform well when there is little overlap in oligomer usage between competing functional classes, but fail if overlap is considerable. Apparently, similar G+C content, a very simple indicator of monomer usage, also indicates substantial overlap in higher order oligomer usage. The G+C dependent performance motivated our quartiled approach, where training data are partitioned by G+C content and a Markov chain is trained for all partitions. Test sequences were first assigned a partition and then classified using partition-specific Markov chains. The deployment of all the models studied under a quartiled framework yielded considerable performance gains in both taxa, but most dramatically in rice (see Table 4).

The quartiled approach involves the estimation of more Markov chains, and presumably far more parameters; the fixed order Markov chain requires four times as many parameters under the quartile methodology, but the interpolated variants automatically adjust the parameter space according to data complexity. However, all methods, including the fixed order Markov chain, classified substantially better under quartile training. The results suggest that enhancing model complexity through chain order may not be the most efficient way to distinguish sequence class. Instead, the assumption that there is a single Markov chain generating each class is suspect. We are currently investigating mixture models where multiple Markov chains generate each class. IMM variants can be seen as mixture models across chain orders, but the resulting parameterization may be overly restrictive.

Our results suggest that use of essentially any of the interpolated estimation methods, coupled with a G+C composition-specific (quartiled) framework, should improve gene annotation in plant genomic sequences, particularly in monocot species, including those with mature (rice) and emerging (maize and sorghum) genomic resources. The availability of our software to efficiently train Markov chains from species-specific and stratified data sets can facilitate incorporation of tailored parameter sets into general *ab initio* gene prediction programs.

Acknowledgements. We would like to thank two anonymous reviewers for helpful suggestions that improved this report. This work was supported in part by NSF Grant DBI-0606909 to V.B. M.E.S was also supported in part by the USDA with an IFAFS Multidisciplinary Graduate Education Training Grant (2001-52100-11506).

References

1. Burge, C., Karlin, S.: Prediction of complete gene structures in human genomic DNA. Journal of Molecular Biology **268** (1997) 78–84
2. Majoros, W., Pertea, M., Antonescu, C., Salzberg, S.: `GlimmerM`, `Exonomy` and `Unveil`: three *ab initio* eukaryotic genefinders. Nucleic Acids Research **31** (2003) 3601–3604
3. Lukashin, A., Borodovsky, M.: `GeneMark.HMM`: new solutions for gene finding. Nucleic Acids Research **26** (1998) 1107–1115
4. Azad, R., Borodovsky, M.: Effects of choice of DNA sequence model structure on gene identification accuracy. Bioinformatics **20** (2004) 993–1005
5. Salzberg, S., Delchur, A., Kasif, S., White, O.: Microbial gene identification using interpolated Markov models. Nucleic Acids Research **26** (1998) 544–548
6. Delcher, A., Harmon, D., Kasif, S., White, O., Salzberg, S.: Improved microbial gene identification with `GLIMMER`. Nucleic Acids Research **27** (1999) 4636–4641
7. Potamianos, G., Jelinek, F.: A study of n-gram and decision tree letter language modeling methods. Speech Communication **24** (1998) 171–192
8. `IMMpractical`. http://sourceforge.net/projects/immpractical/
9. TAIR: The *Arabidopsis* Information Resource. http://www.arabidopsis.org/
10. TIGR: The Institute for Genomic Research. http://www.tigr.org/
11. Zhang, M.: Computational prediction of eukaryotic protein-coding genes. Nature Reviews Genetics **3** (2000) 698–709
12. van Baren, M., Brent, M.: Iterative gene prediction and pseudogene removal improves genome annotation. Genome Research **16** (2006) 678–685
13. TIGR XML Specification. ftp://ftp.tigr.org/pub/data/DTDs/tigrxml.dtd
14. Florea, L.: Bioinformatics of alternative splicing and its regulation. Briefings in Bioinformatics **7** (2006) 55–69
15. Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D.: Basic local alignment search tool. Journal of Molecular Biology **215** (1990) 403–410

16. Borodovsky, M., McIninch, J.: GENMARK: Parallel gene recognition for both DNA strands. Computers in Chemistry **17** (1993) 123–133
17. Salzberg, S., Pertea, M., Delchur, A., Gardner, M., Herve, T.: Interpolated Markov models for eukaryotic gene finding. Genomics **59** (1999) 24–31
18. Sparks, M., Brendel, V.: Incorporation of splice site probability models for non-canonical introns improves gene structure prediction in plants. Bioinformatics **21** (2005) iii20–iii30
19. Baldi, P., Brunak, S., Chauvin, Y., Andersen, C., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. Bioinformatics **16** (2000) 412–424
20. Sing, T., Sander, O., Beerenwinkel, N., Lengauer, T.: ROCR: visualizing classifier performance in R. Bioinformatics **21** (2005) 3940–3941
21. Guigó, R., Brent, M.: Recent advances in gene structure prediction. Current Opinion in Structural Biology **14** (2004) 264–272
22. Siepel, A., Haussler, D.: Computational identification of evolutionarily conserved exons. In: Proceedings of the 8th Annual International Conference on Research in Computational Biology. (2004) 177–186
23. Majoros, W., Pertea, M., Salzberg, S.: Efficient implementation of a generalized pair Hidden Markov model for comparative gene finding. Bioinformatics **21** (2005) 1782–1788
24. Chen, L., DeVries, A., Cheng, C.H.: Convergent evolution of antifreeze glycoproteins in Antarctic notothenioid fish and Arctic cod. Proceedings of the National Academy of Sciences, USA **94** (1997) 3817–3822
25. Roelofs, W., Liu, W., Hao, G., Jiao, H., Rooney, A., Linn Jr., C.: Evolution of moth sex pheromones via ancestral genes. Proceedings of the National Academy of Sciences, USA **99** (2002) 13621–13626

Predicting Palmitoylation Sites Using a Regularised Bio-basis Function Neural Network

Zheng Rong Yang

School of Engineering, Computer Science and Mathematics
University of Exeter, UK

Abstract. Palmitoylation is one of the most important post-translational modifications involving molecular signalling activities. Two simple methods have been developed very recently for predicting palmitoylation sites, but the sensitivity (the prediction accuracy of palmitoylation sites) of both methods is low ($< 65\%$). A regularised bio-basis function neural network is implemented in this paper aiming to improve the sensitivity. A set of protein sequences with experimentally determined palmitoylation sites are downloaded from NCBI for the study. The protein-oriented cross-validation strategy is used for proper model construction. The experiments show that the regularised bio-basis function neural network significantly outperforms the two existing methods as well as the support vector machine and the radial basis function neural network. Specifically the sensitivity has been significantly improved with a slightly improved specificity (the prediction accuracy of non-palmitoylation sites).

Keywords: Palmitoylation site prediction, bio-basis function, regularisation.

1 Introduction

Palmitoylation is a hydrophobic protein-modification activity where fatty acids are covalently attached to cysteine residues of membrane proteins. In biochemistry and enzymology study, it has been observed that this hydrophobic protein-modification activity uses cellular and viral membrane proteins for signal transmission [1]. It is still unknown what the molecular signals for palmitoylation are. Although palmitoylation is known to be a reversible activity with cycles of acylation and deacylation, the relevant enzymatic mechanism has not been completely known because some palmitated proteins are found without any enzyme source present. Despite of these observations, palmitoylation activity has been widely studied in various areas including most signalling pathway activities [2], [3]. For instance, Smotrys et al showed that most trafficking and protein-protein interactions as well as enzyme activities depend on the existence of palmitated proteins [4]. They also showed that palmitated proteins can enhance the membrane interactions and the reversibility of palmitoylation is an attractive mechanism for regulating protein activity and cell signalling. Li and Yang have found that most palmitoylation-deficient mutant Env proteins are

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 406–417, 2007.

soluble when extracted by ice-cold TX-100 and stay at the bottom of the gradients in their study of the association between the Maurine leukaemia virus Env protein and lipid rafts [5]. Palmitoylation has also been studied in disease-related subjects. For instance, Yu and Lee have found that two cysteine residues (257 and 261) at the C-terminal of NS4B have lipid modifications (palmitoylation) in studying the polymerization of Hepatitis C virus NS4B protein [6]. They concluded that site-specific mutagenesis of these cysteine residues are important for protein-protein interactions in the formation of HCV RNA replication complex. Another example of disease-related biology study of palmitoylation activity is vacuolar events. Peng and Tang showed that palmitoylation targets Vac8p to specific membrane sub-domains for vacuole homotypic fusion at three cysteine residues (4, 5 and 7) [7].

Specificity study of post-translational modifications like phosphorylation, methylation, sumoylation and palmitoylation is a very important subject in systems biology research for understanding how proteins are responding to extracellular cues for information transmission along signalling pathways. One of the important subjects in studying post-translational modifications is to identify where the modifications are or where proteins are binding for the modifications. For this kind of study, it is generally not necessary to view whole protein sequences. Rather, one normally focuses on a small area of a binding site or a few residues around a functional site. This study is commonly termed as protein functional site prediction which involves the use of a set of peptides (short regions of protein sequences) with known functional status, i.e. functional or non-functional. In this context, a functional peptide is the one with a palmitoylation site.

The earliest work on protein functional site prediction were normally based on frequency estimate. For example, the h function [8], where the frequency of 20 amino acids at each residue is calculated from a set of functional peptides. The estimated frequencies are then stored in a computer program for prediction. The major shortcoming of this method is that they usually result in high sensitivity and low specificity. Some statistical models like hidden Markov models (HMM) [9], discriminant analysis [10] and quadratic discriminant analysis [11] have also been used for data mining protein peptides. However, a HMM model also has a high sensitivity and a low specificity [12].

Neural networks and the support vector machine [13], [14] have been applied to data mining protein peptides as well. For instance, neural networks have been used in signal peptide cleavage site prediction [15], glycoproteins linkage site prediction [16], enzyme active site prediction [17], phosphorylation site prediction [18], and water active site prediction [19]. The support vector machine has been used for the prediction of translation initiation sites [20], the prediction of phosphorylation sites [21], the prediction of T-cell receptor [22], and the prediction of protein-protein interactions [23].

In the context of predicting palmitoylation sites, Zhou et al first employed a clustering and scoring strategy to build a model to predict palmitoylation sites in early 2006 [24]. In the same group, Xue et al employed a Naive Bayes method to predict palmitoylation sites in late 2006 [25]. They have used 105 protein sequences with 245 palmitoylation sites being experimentally determined. Based on these protein sequences, 977 non-palmitoylation peptides with various lengths were generated, each having a cysteine in the middle. Their model [25] is able to make total prediction accuracy 86.74% with the sensitivity 58.37%, the specificity 93.86% and the Matthews' correlation coefficient 0.5618. In comparison, they have used the support vector machine and the radial basis function neural network. For the former, the specificity is 94.47%, the sensitivity is 64.49% and the Matthews' correlation coefficient is 0.623. For the latter, the sensitivity is 95.09%, the specificity is 55.51% and the Matthews' correlation coefficient is 0.5664. It can be seen that all have a low sensitivity from 58.37% to 64.49%. For the Naive Bayes method, they found that the optimal window size is six. For the support vector machine, they found that the optimal window size is seven. For the radial basis function neural network, they found that the best window size is eight. In dealing with amino acids, they employed the orthogonal coding mechanism where each amino acid is coded using a 20-bit long orthogonal binary vector [26].

Although the orthogonal coding mechanism has been widely used for various protein peptide modelling tasks, it may not well code biological information in peptides. The bio-basis function neural network was therefore developed for proper coding of amino acids in 2003 [27], [28]. The bio-basis function neural network has been successfully used for Trypsin cleavage site prediction [27], HIV cleavage site prediction [28], [30], [29], disordered protein prediction [31], [32], phosphorylation site prediction [33], [12], glycoprotein O-linkage site prediction [34], Caspase cleavage site prediction [35], SARS-CoV protease cleavage site prediction [36], signal peptide prediction [37], [38], and T-cell epitope prediction [39]. In all these applications, no regularisation was applied. This means that the models may possibly overfit to the training peptides. With the regularisation theory [40], we can constrain the model parameters to trade off between bias and variance so as to improve model generalisation capability when a proper regularisation constant is determined. This has been widely studied in neural network community [41], [42].

In this study, a regularised bio-basis function neural network is implemented for improving palmitoylation site prediction sensitivity using the data downloaded from NCBI. First, the regularised bio-basis function neural network is introduced and then how data downloaded from NCBI are organised for simulation is discussed. Particularly, the protein-oriented cross-validation strategy is discussed for proper model construction. A comparison will be given showing if the regularised bio-basis function neural network can improve the sensitivity for palmitoylation site prediction.

2 Regularised Bio-basis Function Neural Network

Before discussing the regularised bio-basis function neural network, the bio-basis function neural net is briefly discussed. Given two peptides $\mathbf{s}_i$ and $\mathbf{s}_j$, the likelihood that they are from the same ancestor through evolution is $\mathcal{L} = \prod_{k=1}^{d} p(s_{ik}, s_{jk})$. Here d is the number of residues in two peptides, $p(s_{ik}, s_{jk})$ is the probability that both s_{ik} and s_{jk} occur in $\mathbf{s}_i$ and $\mathbf{s}_j$ at the same time. Applying a logarithm operation on the likelihood function leads to

$$\rho(\mathbf{s}_i, \mathbf{s}_j) = \ln \mathcal{L} = \sum_{k=1}^{d} \mathcal{M}(s_{ik}, s_{jk}) \tag{1}$$

Here $\mathcal{M}(s_{ik}, s_{jk})$ can be found from various mutation matrices [43], [44], [45]. The bio-basis function is designed as follows

$$\phi(\mathbf{s}_i, \mathbf{s}_j) = \exp\left(\frac{\rho(\mathbf{s}_i, \mathbf{s}_j) - \rho(\mathbf{s}_j, \mathbf{s}_j)}{\rho(\mathbf{s}_j, \mathbf{s}_j)}\right) \tag{2}$$

It can be seen that $\phi(\mathbf{s}_i, \mathbf{s}_j) \in (0, 1]$. When $\mathbf{s}_i = \mathbf{s}_j$, $\phi(\mathbf{s}_i, \mathbf{s}_j) = 1$. When $\mathbf{s}_i$ is very different from $\mathbf{s}_j$, $\phi(\mathbf{s}_i, \mathbf{s}_j) \to 0$. By using the log-odds-ratio, a model using the bio-basis function for peptide classification is defined as

$$y_n = \frac{1}{1 + \exp\left(-\mathbf{w}^T \phi_n\right)} \tag{3}$$

Here $\mathbf{w} = (w_0, w_1, \cdots, w_\ell)^T$ and $\phi_n = (1, \phi_{n1}, \phi_{n2}, \cdots, \phi_{n\ell})^T$. The objective function with an added regularisation term (neg log pdf + regularisation) is then defined as

$$\mathcal{O} = -\sum_{n=1}^{\ell} \left(t_n \log y_n + (1 - t_n) \log(1 - y_n)\right) + \frac{1}{2}\lambda \mathbf{w}^T \mathbf{w} \tag{4}$$

Here λ is a regularisation constant. The update rule of the parameters is defined as

$$\Delta \mathbf{w} = -\mathbf{H}^{-1} \nabla \mathcal{O} = \left(\mathbf{\Phi}^T \mathbf{\Lambda} \mathbf{\Phi} + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{\Phi}^T \mathbf{e} - \lambda \mathbf{w}\right) \tag{5}$$

Here $\mathbf{I}$ is an identity matrix, $\mathbf{\Lambda} = \mathrm{diag}\{y_n(1 - y_n)\}$ is called an entropy matrix, $\nabla \mathcal{O}$ is the first derivative of $\mathcal{O}$ with respect to $\mathbf{w}$,

$$\mathbf{H} = \nabla\nabla\mathcal{O} \tag{6}$$

is the Hessian matrix, $\mathbf{e} = (e_1, e_2, \cdots, e_\ell)^T$, $e_n = t_n - y_n$, and

$$\mathbf{\Phi} = \begin{pmatrix} 1 & \phi_{11} & \phi_{12} & \cdots & \phi_{1\ell} \\ 1 & \phi_{21} & \phi_{22} & \cdots & \phi_{2\ell} \\ 1 & \vdots & \vdots & \vdots & \vdots \\ 1 & \phi_{\ell 1} & \phi_{\ell 2} & \cdots & \phi_{\ell\ell} \end{pmatrix} \tag{7}$$

The learning procedure is designed as below

[1]: $c = 0$, $\mathbf{w}^c = \mathbf{0}$
[2]: Calculate $\mathbf{y}^c = (y_1^c, y_2^c, \cdots, y_\ell^c)^T$, $\mathbf{e}^c$ and $\mathbf{\Lambda}^c$

[3]: $\mathbf{w}^{c+1} = \mathbf{w}^c + \left(\mathbf{\Phi}^T \mathbf{\Lambda}^c \mathbf{\Phi} + \lambda \mathbf{I}\right)^{-1} \left(\mathbf{\Phi}^T \mathbf{e}^c - \lambda \mathbf{w}^c\right)$
[4]: If $\|\mathbf{w}^{c+1} - \mathbf{w}^c\| < \epsilon$, stop, otherwise $c = c + 1$, goto [2].

Here $\mathbf{w}^c$ is $\mathbf{w}$ at c^{th} learning cycle, $\mathbf{y}^c$ is $\mathbf{y}$ at c^{th} learning cycle, $\mathbf{e}^c$ is $\mathbf{e}$ at c^{th} learning cycle, $\mathbf{\Lambda}^c$ is $\mathbf{\Lambda}$ at c^{th} learning cycle and $\epsilon > 0$ is a small number functioning as a termination rule. The other termination rule is the maximum training cycle (being set 100 in this paper). In most cases, the first termination rule is satisfied.

3 Result

3.1 Data

A data set of 55 protein sequences with 90 experimentally confirmed palmitoylation sites was downloaded from NCBI. It has been found that palmitoylation activity won't happen if cysteine is not present. We can then scan these 55 protein sequences to generate peptides with cysteine in the middle (P_0). In total, there are 490 cysteine residues in these 55 protein sequences. There are on average 8.9 cysteine residues in each protein sequence and less than two of them are possible palmitoylation sites. The peptide chain with $2n+1$ residues is expressed as $\text{P}_n - \cdots - \text{P}_1 - \text{P}_0 - \text{P}_{1'} - \cdots - \text{P}_{n'}$.

3.2 Cross-Validation

The next question is then how to use the data for constructing models for prediction. We don't want a model to overfit the training data in any circumstances. Cross-validation or jackknife is certainly a way for achieving this goal. However, the fundamental principle of cross-validation has been very often abused in data mining protein peptides. In using cross-validation, one important principle is that we cannot use a data set which has any information exposed to training for evaluation. If this happens, the model can be very likely over-evaluated. However, in many applications, this important issue has not been seriously addressed. Subsequences or peptides are normally generated through scanning all the available protein sequences at first. These generated peptides are then pooled together and then randomly divided for cross-validation. With this strategy, we may almost over-evaluate a model or an algorithm. The reason is very simple. Each protein sequence can be treated as a small world in which mutation (although the underline mechanism of it has not yet been completely known) happens in a specific way which may differ from other protein sequences. If we have generated peptides before cross-validation, some peptides generated from a protein sequence can be randomly picked up for training and some peptides generated from the same protein sequence can be randomly picked up for testing. This means that the pattern in testing peptides have already partially known in training! To handle this problem, we have proposed a new strategy called protein-oriented cross-validation [49], [50]. The core principle of the protein-oriented cross-validation is to divide protein sequences into k folds at first. For each sequence in each fold, a sliding window is applied to generate peptides. Cross-validation simulation is then run based on peptides in these folds.

3.3 Sequence Logos

Fig 1 and 2 show the sequence logos for palmitated and non-palmitated peptides. The logos were produced using the WebLogo[1] [47]. Note that the middle cysteine residue is removed. This means that we are working for the peptides in the following format $P_n - \cdots - P_1 - P_{1'} - \cdots - P_{n'}$. From Fig 1 and 2, it can be seen that two classes of peptides show some difference in amino acid distributions.

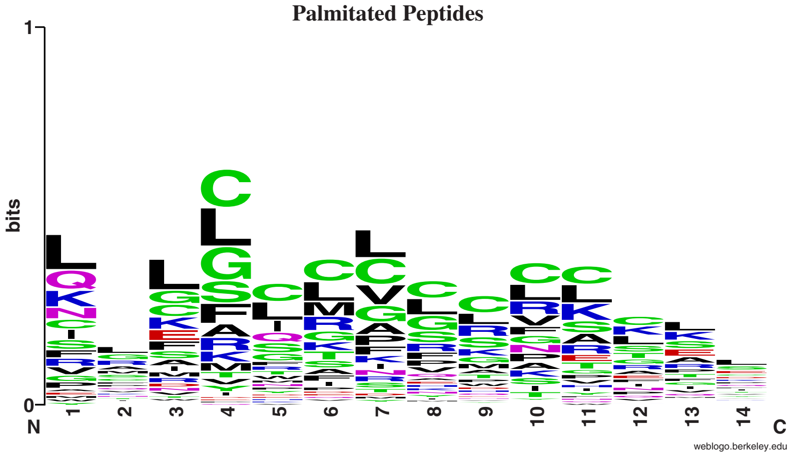


Fig. 1. Sequence logos generated for palmitated peptides

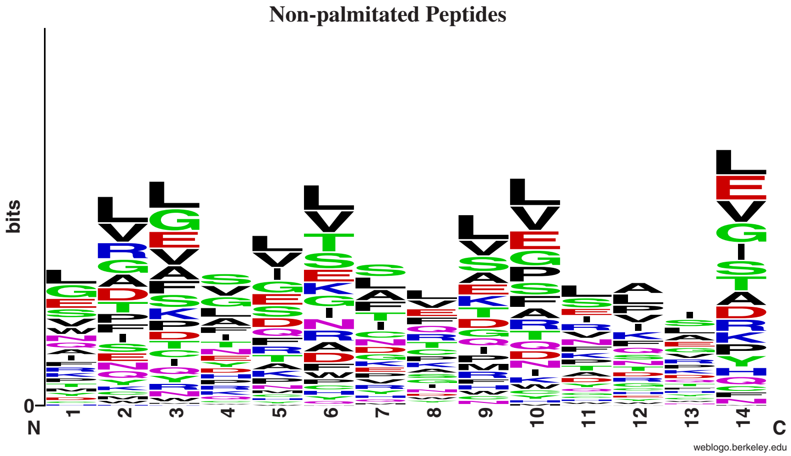


Fig. 2. Sequence logos generated for non-palmitated peptides

[1] http://weblogo.berkeley.edu/logo.cgi

3.4 Model Evaluation

Two criteria are used for model evaluation. They are the Matthews' correlation coefficient [46] and the receiver operating characteristic (ROC) curve [48]. Let TN, TP, FN, FP denote true negative (correctly identified non-palmitated peptides), true positive (correctly identified palmitated peptides), false negative (palmitated peptides identified as non-palmitated ones) and false positive (non-palmitated peptides identified as palmitated ones), respectively. The Matthews' correlation coefficient (MCC) is

$$MCC = \frac{TN \times TP - FN \times FP}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{8}$$

The Matthews' correlation coefficient measures how the predictions correlate with the real target values. If the coefficient is positive, the predictions are positively correlated with the target values. If the Matthews' correlation coefficient is zero, the prediction is completely random. For the ROC analysis, we use the area under a ROC curve (AUR) for the testing set as it is a quantitative measurement of the robustness of a built model.

3.5 Result

For simulation, we have changed the λ (see Eq. 4) value from the range (0.001, 0.002, 0.004, 0.006, 0.008, 0.01, 0.02, 0.04, 0.06, 0.08). The simulation result using 10 λ values for 6 window sizes (5, 7, 9, 11, 13, and 15) are shown in Fig 3

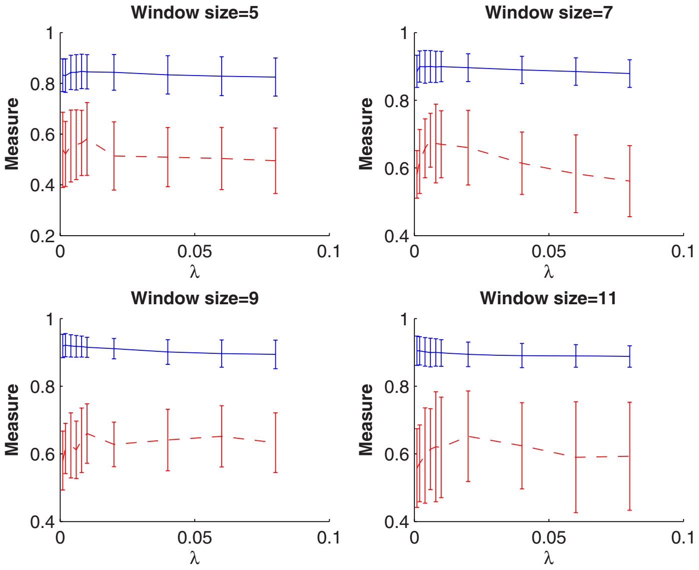


Fig. 3. Performance for window sizes 5, 7, 9, and 11

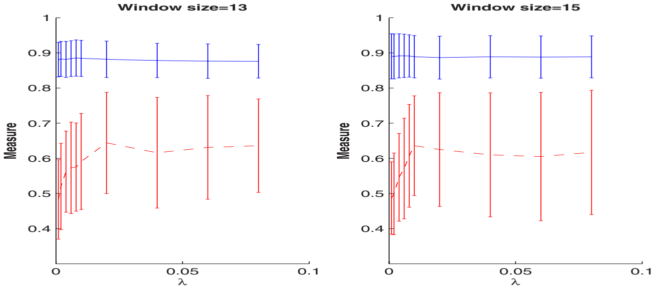
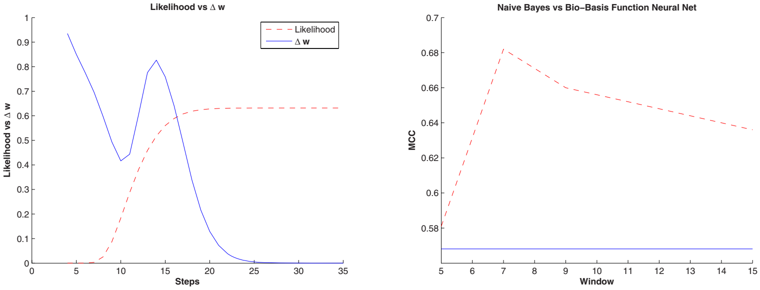


Fig. 4. Performance for window sizes 13 and 15



(a) (b)

Fig. 5. (a) Learning performance using the regularised bio-basis function neural network. (b) MCC comparison between Naive Bayes and Bio-Basis Function Neural Net.

and Fig 4, where the dashed lines are for MCC and the solid lines are for AUR. It can be seen for almost all cases, small λ values produce better models.

Fig 5 (a) shows the learning performance using the regularised bio-basis function neural network. It can be seen that the likelihood $\prod_{n=1}^{\ell} y_n^{t_n}(1-y_n)^{1-t_n}$ is consistently increasing and the change of weights $\|\varDelta\mathbf{w}\|$ is decreasing.

3.6 Comparison

Table 1 shows a comparison between different algorithms. It can be seen that the regularised bio-basis function neural net significantly outperforms the Naive Bayes, the support vector machine and the radial-basis function neural net. The best bio-basis function neural network uses 7-mer (in fact 6-mer after removing P_0) peptides and $\lambda = 0.006$. The best bio-basis function neural net and the Naive Bayes show a difference 0.12 in MCC. The best bio-basis function neural net and the Naive Bayes show a difference 0.09% in specificity. However, the best bio-basis function neural net and the Naive Bayes show a difference 16.04% in sensitivity accounting for 27.5% increase! Fig 5 (b) shows a MCC comparison between the Naive Bayes and the bio-basis function neural network, where the dash line represents the bio-basis function neural network using various window sizes while the straight solid line represents the performance using the Naive Bayes method. It can be seen that the latter much outperforms the former.

It has been mentioned above that the best window sizes are six using the Naive Bayes method, seven using the support vector machine, eight using the radial-basis function neural network. The regularised bio-basis function neural network selects the best window size as seven as the support vector machine which produced the best sensitivity compared with the rest conventional methods. It should be noted that both the support vector machine and the regularised bio-basis function neural network use the regularisation theory to improve generalisation capability. This may be the reason that these two are close in the prediction sensitivity and the window size.

Table 1. The comparison between different algorithms. Naive Bayes, Support vector machine and Radial-basis function neural net have no report of AUR measures. The number within brackets mean the window size used for modelling.

Algorithms	λ	Specificity	Sensitivity	MCC	AUR
Naive Bayes		93.86%	58.37%	0.562	n.a.
Support vector machine		94.47%	64.44%	0.623	n.a.
Radial-basis function neural net		95.09%	55.51%	0.537	n.a.
Bio-basis function neural net (5)	0.01	89.78%	70.10%	0.581	0.845
Bio-basis function neural net (7)	0.006	95.95%	74.41%	0.682	0.899
Bio-basis function neural net (9)	0.01	93.70%	72.53%	0.660	0.920
Bio-basis function neural net (11)	0.02	95.41%	63.39%	0.652	0.894
Bio-basis function neural net (13)	0.02	93.66%	68.67%	0.644	0.882
Bio-basis function neural net (15)	0.01	89.88%	77.24%	0.636	0.889

It should be noted that the models presented by Zhou et al [24] and Xue et al [25] used 245 palmitated peptides and 977 non-palmitated peptides compared with 90 palmitated peptides and 400 non-palmitated peptides. The author is contacting Zhou et al and Xue et al at the moment for requesting their data. It is expected that the regularised bio-basis function neural network will even perform better after their data arrive.

4 Conclusion

This paper has implemented a regularised bio-basis function neural network for predicting palmitoylation sites in proteins. Through comparison, it has been found that the new method presented in this paper significantly outperforms the traditional methods, namely the Naive Bayes method, the support vector machine and the radial-basis function neural network. We are currently investigating how to use the regularised bio-basis function neural network to produce sparse models for better interpretation to the trained models. In using the information provided by the Hessian matrix, we can evaluate the importance of each bio-basis using the statistic as $Z_n = \frac{w_n}{\sqrt{H^{-1}_{nn}}}$ $(\forall n \in [0, \ell])$. If $Z_n < \vartheta$ $(\vartheta > 0$

is a small number functioning as a threshold), w_n can be zeroed or the n^{th} biobasis can be removed. A detailed research is undergoing for investigating how to determine ϑ and how to interpret the left bio-bases in biology.

References

1. Veit, M., Schmidt, M.F.G.: Palmitoylation of viral and cellular proteins. In: Influenza Viruses. Facts and Perspectives, Schmidt, Michael F.G.(Hrsg.) Berlin: Grosse-Verlag ISBN: 3-9810221-3-0 (2006)
2. Navarro-Lerida, I., Alvarez-Barrientos, A., Rodrguez-Crespo, I.: N-terminal palmitoylation within the appropriate amino acid environment conveys on NOS2 the ability to progress along the intracellular sorting pathways. Journal of Cell Science **119** (2006) 1558–1596
3. Kurayoshi, M., Yamamoto, H., Izumi, S., Kikuchi, A.: Post-translational palmitoylation and glycosylation of Wnt-5a are necessary for its signaling
4. Smotrys, J.E., Linder, M.E.: Palmitoylation of intracellular signaling proteins: regulation and function. Annu. Rev. Biochem. **73** (2004) 559–587
5. Li, M., Yang, C., Tong, C., Weidmann, A., Compans, R.W.: Palmitoylation of the murine leukemia virus envelope protein is critical for lipid raft association and surface expression. J Virol. **76** (2002) 11845–11852
6. Yu, G., Lee, K., Gao, L., Lai, M.M.C.: Palmitoylation and Polymerization of Hepatitis C Virus NS4B Protein. Journal of Virology **80** (2006) 6013–6023
7. Peng, Y., Tang, F., Weisman, L.S.: Palmitoylation plays a role in targeting Vac8p to specific membrane subdomains. Traffic **7** (2006) 1378
8. Poorman, R.A., Tomasselli, A.G., Heinrikson, R.L., Kezdy, F.J.: A cumulative specificity model for protease from human immunodeficiency virus types 1 and 2, inferred from statistical analysis of an extended substrate data base. J. Biol. Chem **22** (1991) 14554–14561
9. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77** (1989) 257–286
10. Nakata, K., Maizel, J.V.: Prediction of operator-binding protein by discriminant analysis. Gene Anal Tech **6** (1989) 111–119
11. Chen, C.P., Rost, B.: State-of-the-art in membrane protein prediction. Applied Bioinformatics **1** (2002) 21–35
12. Senawongse, P., Dalby, A., Yang, Z.R.: Predicting the phosphorylation sites using hidden Markov models and Machine Learning methods. Journal of Chemical Information and Computer Science **45** (2005) 1147–1152
13. Vapnik, V.: The Nature of Statistical Learning Theory.Springer-Verlag, New York (1995)
14. Scholkopf, B.: The kernel trick for distances, Technical Report. Microsoft Research May (2000)
15. Nielsen, M., Lundegaard, C., Worning, P., Lauemoller, S.L., Lamberth, K., Buss, S., et al. Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. Protein Science **12** (2003) 1007–1017
16. Hansen, J.E., Lund, O., Engelbrecht, J., Bohr, H., Nielsen, J.O.: Prediction of O-glycosylation of mammalian proteins: specificity patterns of UDP-GalNAc: polypeptide N-acetylgalactosaminyltransferase. Biochem J. **30** (1995) 801–813
17. Gutteridge, A., Bartlett, G.J., Thornton, J.M.: Using a neural network and spatial clustering to predict the location of active sites in enzymes. Journal of Molecular Biology **330** (2003) 719–734

18. Blom, N., Gammeltoft, S., Brunak, S.: Sequence and structure based prediction of eukaryotic protein phosphorylation sites. J. Mol. Biol **24** (1999) 1351–1362
19. Ehrlich, L., Reczko, M., Bohr, H., Wade, R.C.: Prediction of protein hydration sites from sequence by modular neural networks. Protein Eng **11** (1998) 11–19
20. Zien, A., Ratsch, G., Mika, S., Scholkopf, B., Lengauer, T. and Muller, K.R.: Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics **16** (2000) 799–807
21. Kim, J.H., Lee, J., Oh, B., Kimm, K., Koh, I.: Prediction of phosphorylation sites using SVMs. Bioinformatics **20** (2006) 3179–3184
22. Zhao, Y., Pinilla, C., Valmori, D., Martin, R., Simon, R.: Application of support vector machines for T-cell epitopes prediction. Bioinformatics **19** (2003) 1978–1984
23. Koike, A., Takagi, T.: Prediction of protein-protein interaction sites using support vector machines. Protein Eng Des Sel **17** (2004) 165–173
24. Zhou, F., Xue, Y., Yao, X., Xu, Y.: CSS-Palm: palmitoylation site prediction with a clustering and scoring strategy (CSS). Bioinformatics **22** (2006) 894–896
25. Xue, Y., Chen, H., Jin, C., Sun, Z., Yao, X.: NBA-Palm: prediction of palmitoylation site implemented in Nave Bayes algorithm. BMC Bioinformatics **7** (2006) 1–10
26. Qian, N., Sejnowski, T.: Predicting the secondary structure of globular proteins using neural network models. Proceeding of Int J. Conf. On Neural Networks, (1998) 865–884
27. Thomson, R., Hodgman, T., Yang, Z.R., Doyle, A.: Characterising proteolytic cleavage site activity using bio-basis function neural networks. Bioinformatics **19** (2003) 1741–1747
28. Yang, Z.R., Thomson, R.: Bio-basis function neural network for prediction of protease cleavage sites in proteins. IEEE Trans. on Neural Networks **16** (2005) 263–274
29. You, L., Garwicz, D., Rognvaldsson, T.: Comprehensive bioinformatic analysis of the specificity of human immunodeficiency virus type 1 protease. Journal of Virology **79** (2005) 12477–12486
30. Yang, Z.R., Berry, E.: A novel neural learning algorithm for protease cleavage site prediction. Journal of Bioinformatics and Computational Biology **2** (2004) 511–531
31. Thomson, R., Esnouf, R.: Predict disordered proteins using bio-basis function neural networks. Lecture Notes in Computer Science **3177** (2004) 19–27
32. Yang, Z.R., Thomson, R., McNeil, P., Esnouf, R.: RONN: use of the bio-basis function neural network technique for the detection of natively disordered regions in proteins. Bioinformatics **21** (2005) 3369–3376
33. Berry, E., Dalby, A., Yang, Z.R.: Reduced bio-basis function neural networks in prediction of phosphorylation sites, a comparative study. Computational Biology and Chemistry **28** (2004) 75–85
34. Yang, Z.R., Chou, K.C.: Predicting the O-linkage sites in glycoproteins using bio-basis function neural networks. Bioinformatics **20** (2004) 903–908
35. Yang, Z.R.: Prediction of caspase cleavage sites using Bayesian bio-basis function neural networks. Bioinformatics **21** (2005) 1831–1837
36. Yang, Z.R.: Mining SARS-CoV protease cleavage data using decision trees, a novel method for decisive template searching. Bioinformatics **21** (2005) 2644–2650
37. Sidhu, A., Yang, Z.R.: Prediction of signal peptides using bio-basis function neural networks and decision trees. Applied Bioinformatics **5** (2006) 13–19
38. Yang, Z.R.: Orthogonal kernel machine in prediction of functional sites in preteins. IEEE Trans on Systems, Man and Cybernetics **35** (2005) 100–106

39. Yang, Z.R., Johnathan, F.: Predict T-cell epitopes using bio-support vector machines. Journal of Chemical Information and Computer Sciences **45** (2005) 1142–1148
40. Neumaier, A.: Solving ill-conditioned and singular linear systems: A tutorial on regularization, SIAM Review **40** (1998) 636–666
41. Girosi, F., Jones, M., Poggio, T.: Regularization Theory and Neural Networks Architectures Neural Computation **7** (1995) 219–269
42. Bishop, C.: Neural Networks for Pattern Recognition, Oxford Press, 1995
43. Dayhoff, M.O., Schwartz, R.M., Orcutt, B.C.: A model of evolutionary change in proteins. matrices for detecting distant relationships. Atlas of protein sequence and structure **5** (1978) 345–358
44. Henikoff, S., Henikoff, J.G.: Amino acid Substitution matrices from protein blocks. Proc. Natl. Acad. Sci. **89** (1992) 10915–10919
45. Johnson, M.S., Overington, J.P.: A structural basis for sequence comparisons-an evaluation of scoring methodologies. Journal Molecular Biology **233** (1993) 716–738
46. Matthews, B.W.: Comparison of the predicted and observed secondary structure of T4 phage lysozyme. Biochim Biophys Acta **405** (1975) 442–451
47. Schneider, T.D., Stephens, R.M.: Sequence Logos: A new way to display consensus sequences. Nucleic Acids Res. **18** (1990) 6097–6100
48. Metz, C.E.: Basic principles of ROC analysis. Seminars in Nuclear Medicine **8** (1978) 283–298
49. Yang, Z.R.: Predicting Hepatitis C virus protease cleavage sites using generalised linear indicator regression models. IEEE Trans on Biomedical Engineering. **53** (2006) 2119–2123
50. Yang, Z.R.: A probabilistic peptide machine for predicting Hepatitis C virus protease cleavage sites. IEEE Trans on Information Technology in Biomedicine (in press)

A Novel Kernel-Based Approach for Predicting Binding Peptides for HLA Class II Molecules

Hao Yu[1], Minlie Huang[1], Xiaoyan Zhu[1,*], and Yabin Guo[2]

[1] Department of Computer Science and Technology, Tsinghua University
th@mails.tsinghua.edu.cn, {aihuang,zxy-dcs}@tsinghua.edu.cn
[2] School of Biomedicine, Ministry of Education Key Laboratory of Bioinformatics,
Tsinghua University, Beijing 100084, P. R. China
gyb06@mails.tsinghua.edu.cn

Abstract. Peptides that bind to Human Leukocyte Antigens (HLA) can be presented to T-cell receptor and trigger immune response. Identification of specific binding peptides is critical for immunology research and vaccine design. However, accurate prediction of peptides binding to HLA molecules is challenging. A variety of methods such as HMM and ANN have been applied to predict peptides that can bind to HLA class I molecules and therefore the number of candidate binders for experimental assay can be largely reduced. However, it is a more complex process to predict peptides that bind to HLA class II molecules. In this paper, we proposed a kernel-based method, integrating the BLOSUM matrix with string kernel to form a new kernel. The substitution score between amino acids in BLOSUM matrix is incorporated into computing the similarity between two binding peptides, which exhibits more biological meaning over traditional string kernels. The promising results of this approach show advantages than other methods.

Keywords: SVM, string kernel, kernel-based, HLA.

1 Introduction

Major Histocompatibility Complex (MHC) molecules are cell surface glycoproteins presented on antigen presenting cells. They recognize and bind peptides and then trigger immune response. MHC molecules derived from human alleles are called Human Leukocyte Antigens (HLA), which are important for designing vaccines and developing immunotherapies against cancer and autoimmune diseases[1]. As an example, HLA-DRB1*0401 is reported to associate with rheumatoid arthritis[2]. Due to the cost limitation of traditional experimental methods, it is necessary to exploit efficient in silico methodologies to predict candidate binding peptides to reduce the number of experiments required for identifying helper T cell epitopes.

HLA genes have different alleles and are therefore highly polymorphic. More than 1800 different HLA molecules have been reported[3], and they can be divided into two classes, HLA class I and HLA class II, both of which have a groove for binding

* The corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 418–429, 2007.

peptides[4]. Unlike HLA class I molecules, the groove of HLA class II molecules is open at both ends and is not well defined yet. Previous work suggested that HLA class I molecules can hold peptides of lengths ranging from 8 to 10 amino acids[5], and length of HLA class II binding peptides could vary from 13 to 25 residues[4]. It is clear from previous studies that the prediction of binding peptides of HLA II molecules is much more complex mainly due to 1) the various lengths of reported binding peptides, 2) the undetermined core regions for individual peptides, and 3) the number of amino acids permissible as primary anchors[6].

In recent years, a large number of predicting methods have been developed to identify HLA class I binding peptides, but only a few for HLA class II molecules. All these approaches can be classified as structure- or sequence-based methods. Structure-based methods take the crystal structure of HLA molecules into account and evaluate the compatibility between an HLA molecule and a peptide[7], and some methods even estimate the binding free energies of a peptide to an HLA molecule[8]. Structure-based methods are promising but require already known HLA structures. More seriously, high computational cost limits its large-scale applications.

Sequence-based approaches include three groups: motifs-based approaches, quantitative matrices and machine learning approaches.

Motif-based approaches[9-11] are the earliest approaches which compute the frequency of specific amino acids at special position to generate sequence patterns, then use the patterns to predict binding peptides. However, these methods achieved high specificity but low sensitivity. Hammer gives a good review[12] for these approaches.

The second type of methods is based on quantitative matrices[13-15]. These methods assume the binding affinity between HLA molecules and peptides can be modeled as the sum of independent residues at each position. Hammer [13] presented quantitative matrices to calculate weights for each amino acid in every position and then computed the binding score of peptides. SYFPEITHI[16] is another frequently used web server for prediction. However, it is difficult to align peptides with various lengths and quantitative matrix based models require a large amount of training data. For some kinds of HLA class II alleles, such methods cannot generate matrix models currently.

Machine learning approaches, such as ANN[1, 6] and HMM[17], using known binding and non-binding peptides with fixed lengths to train models and then predict binding peptides., Machine-learning approaches are expected to be more powerful with the increasing data of HLA binding peptides.

SVM is reported to perform better than any other machine learning prediction methods[18, 19]. HLA-DR4Pred[2], SVMHC[18], and SVRMHC[20] are recently reported SVM-based prediction web servers. However, they only examined three common kernels (linear, Gaussian and polynomial kernels) and still need to extract 9mers from peptides which longer than 9mers.

Most previous methods can only handle a fixed length of peptides (usually 9mers), or have to identify a fixed length of continuous core region prior to binding prediction, where this assumption is still biologically questionable, particularly for the non-binders. Obviously, there is no core region for non-binding peptides. Many alignment or extracting algorithms[21-24] are proposed to identify the core region in recent years. Hammer[13], Brusic[6], HLA-DR4Pred[2], MHCPred[25] have developed different matrixes to identify core regions of peptide that binds to HLA-DRB1*0401. Doytchinova[25] has examined 6 alignment matrixes including the referred ones. Unfortunately, no overall correlation was found between them. Therefore, matrix-based alignment approaches is doubtfully.

In our previous study[26], we reported a string kernel method for predicting HLA class II peptides, which is able to handle binding peptides with various lengths. In this paper, we propose a novel kernel-based approach, which integrates BLOSUM matrix with the previous string kernel method. In our approach, there is no need to extract core regions before binding prediction and a given peptide can be classified as a binder or non-binder directly. By using BLOSUM matrix to measure the substitution similarity between amino acids, our approach outperforms previous methods on predicting benchmark datasets. Comparative results show that our approach is a promising model for the prediction of binding peptides for many HLA class II alleles.

The rest of this paper is structured as follows: in Section 2, we present the method of binding peptides prediction using the proposed kernel method in detail. In Section 3, the datasets and experimental evaluation are stated. In Section 4, we make our conclusions and discussions.

2 Method

This paper presents a kernel-based method to predict binding peptides of HLA class II molecules. Kernel-based methods, most integrated with Support vector machines, are widely used in classification of microarray data, Protein homology detection and other biological problems[27, 28], which has been proved to performed better than other prediction methods.

2.1 String Kernel

A function that calculates the inner product for input examples in a feature space is a kernel function. More formally, is a kernel function for any mapping function from the object space X to a feature space. The kernel computes the inner product by implicitly mapping the examples to the feature space. The mapping function transforms a d-dimensional example x into an N-dimensional feature vector, as follows:

$$\phi(x) = (\phi_1(x), \cdots, \phi_N(x)) = (\phi_i(x)), for\ i = 1, \cdots, N \tag{1}$$

However, the explicit extraction of features in a feature space generally has very high computational cost, and a kernel function provides a way to compute in an implicit feature space. The mathematical foundation of the kernel was established by Mercer[29].

Kernels have been well integrated into the framework of support vector machines, which produces the following decision function:

$$f(x) = \operatorname{sgn}\{\sum_{i=1}^{n} \alpha_i^* y_i K(x_i, x) + b^*\} \tag{2}$$

Traditional SVM learning framework requires input objects are represented in feature vectors, which means the feature space is explicitly constructed. Kernels such as linear kernel, RBF kernel, and polynomial kernel are widely used in the context. However, objects, for example, strings and sequences, are not always easy to represent in a form of feature vector. Kernels for complex objects such as tree kernels and string kernels have been well-studies for text categorization or relation extraction [30] in the communities of natural language processing. As peptides can be viewed as strings, we here adopt string kernel for classifying binding and non-binding peptides for HLA class II molecules. The sequence of a peptide consists of various numbers of amino acids, ranging from 9 to 30. More specifically, a peptide can be viewed as a string whose alphabet consists of twenty amino acids. Given a string $s = s_1s_2s_3...s_{|s|}$, string u is the sub-string of s if there exists an integer sequence $\vec{i} = (i_1,i_2,...,i_{|u|})$ $(1\le i_1 <...<i_{|u|}\le |s|)$ satisfying $u_j=s_{i_j}$ for all j = 1,2,..., $|u|$, which can be denoted by $\vec{i} : s.u \equiv s_{i_1}s_{i_2}\cdots s_{i_{|u|}}$. The length of sub-string $s.u$ is $l(s.u) = i_{|u|}-i_1+1$. The string kernel that computes the similarity between peptide x and y is defined by

$$K_n(x,y)=\sum_{u\in\sum^n}\sum_{\vec{i}:x.u}\sum_{\vec{j}:y.u}\lambda^{l(x.u)+l(y.u)} \tag{3}$$

where $\sum^n$ is all possible substrings of length n defined on alphabet $\sum$, and the alphabet consists of twenty amino acids; λ is a decay factor.

Usually the kernel should be normalized to [0,1] by the below transformation:

$$\widehat{K}_n(x,y)=\frac{K_n(x,y)}{\sqrt{K_n(x,x)*K_n(y,y)}} \tag{4}$$

The kernel essentially calculates the similarity between two peptides, and the SVM classifier makes a prediction by training a hyper-plane implied by supporting peptides.

Example. Consider two short peptides LGE and LEY. If we assume k = 2, there are five unique sub-strings as follows:

	L-G	L-E	G-E	L-Y	E-Y
ϕ(LGE)	λ^2	λ^3	λ^2	0	0
ϕ(LEY)	0	λ^2	0	λ^3	λ^2

The similarity between LGE and LEY is K(LGE, LEY) = λ^5, and K(LGE, LGE) = K(LEY, LEY) = $2\lambda^4+\lambda^6$. Hence the normalised value is $\lambda^5 / (2\lambda^4+\lambda^6) = \lambda / (2+\lambda^2)$. The algorithm has been applied in our previous work [26] and achieved good performance.

2.2 Incorporating BLOSUM Matrix into String Kernel

String kernel works well for many tasks of sequence classification[30, 31]. However, difference among different amino acids is not considered since each amino acid is viewed as a letter equally. We have observed that peptides PK$\underline{\text{V}}$VKQNTLKLAT and

PKIVKQNTLKLAT are both bind to HLA-DRB1*0401[32] while there is only a substitution in position 3. In another case, peptide VRSMAAAAA binds to HLA-DRB1*0401[32] while TRSMAAAAA does not[33]. We noticed that Val (V) and Ile(I) are both aliphatic amino acids, but Thr(T) is hydroxyl. It is a reasonable assumption that the chemical properties of residues may associate with binding affinity. Thus a substitution from V to I does not change the binding affinity but a replacement from V to T does. Traditional string kernel is not able to capture such difference.

Therefore, we incorporate BLOSUM matrix into computing the similarity between two peptides, and propose a new kernel named BLOSUM substitution string kernel (BSSK). The basic idea is that the probability of a substitution from one amino acid to another one should be involved in computation of the similarity between peptides. For example, the similarity between PKVV and PKIV should be larger than that between PKVV and PKTV although the two pairs both have only one different amino acid, because the BLOSUM62 substation score for V↔I is 3, while 0 for V↔T. Our approach works as follows:

First, build a feature vector where each dimension is a sub-string of peptides. Since binding peptides are short (no more than 40), we take the length of sub-string n = 2 here. The dimension of the feature vector is 20*20=400 (20 types of amino acids), and each dimension corresponds to a pair of amino acids.

Second, all sub-strings of a given peptide are enumerated. For instance, given a peptide $X = x_1 x_2 ... x_m$, we have $C_m^{n=2}$ sub-strings.

Third, calculate the weight value for each dimension of the feature vector. Suppose one dimension corresponds to a pair $f_i f_j$, the weight value for this dimension is computed as the below:

$$w(X, f_i f_j) = \sum_{\vec{i}:X.u\&|u|=n} (\lambda^{l(X.u)} * \beta(u \leftrightarrow f_i f_j)) \tag{5}$$

Where Equation $\vec{i}$:$X.u$ & $|u| = n$ means u is an n-length sub-string of peptide X, $u=u_1u_2$, and

$$\beta(u \leftrightarrow f_i f_j) = 2^{s(u_1,f_i)+s(u_2,f_j)-\frac{1}{2}[s(u_1,u_1)+s(u_2,u_2)+s(f_i,f_i)+s(f_j,f_j)]} \tag{6}$$

Where $s(a_1,a_2)$ is the substitution score between amino acid a_1 and a_2 in BLOSUM matrix. The substitution factor $\beta(u \leftrightarrow f_i f_j)$ computes the relative possibility that one peptide changes to another one.

At last, the kernel between two peptides X and Y is the direct inner product of weight vectors:

$$K(X,Y) = \sum_{\forall f_i f_j} w(X, f_i f_j) * w(Y, f_i f_j) \tag{7}$$

Similarly as Equation (4), the kernel is normalized to [0,1].

In comparison to traditional string kernels, we adopt a substitution factor $\beta(u \leftrightarrow f_i f_j)$ from BLOSUM matrix to measure how similar two peptides are although they do not own the same sub-string. This factor can exhibit more biological knowledge and meaning than traditional string kernels.

Example. Consider peptides mentioned early in this section. If we assume $n = 2$ and $\lambda = 0.7$, the kernel values is shown in **Table 1**. Our approach successfully confirms the fact that the similarity between PKVVKQNTLKLAT and PKIVKQNTLKLAT is larger.

Table 1. Kernel values calculated with two different algorithms

	StringKernel	our approach
K(PKVVKQNTLKLAT, PKIVKQNTLKLAT)	0.90	0.97
K(VRSMAAAAA, TRSMAAAAA)	0.96	0.96

2.3 Implementation and Parameter Optimization

We implement our algorithm with the aid of SVMLight[34] package. Here binary classification is performed to separate binding peptides (with class label +1) from non-binding ones (with class label -1). Peptides do not need be aligned to extract a core region before classification because our approach offers computational ability for various lengths of peptides.

As our previous work[26], Each peptide is represented as a string sequence in which one character stands for an acid amino. There are still two important parameters in the method, namely the length of sub-string n and the decay factor λ. To measure the performance of parameter optimization we carry out 5-folder cross validation and use Matthews Correlation coefficients to measure the performance.

3 Experiment

3.1 Peptides Dataset

Different methods for MHC binding peptide prediction are hard to compare because most of them are tested on very small test datasets or unpublished datasets. Therefore, Our datasets contain peptides with experimentally natural binders or non-binders for HLA-DRB1*0401 allele. The data are extracted from MHCBench (URL: http://www.imtech.res.in/raghava/mhcbench/), a major database that offers benchmark datasets for evaluating the methods for predicting HLA class II binding peptides. Choosing MHCBench is because MHCBench lists some comparable results from some previously reported methods and choosing HLA-DRB1*0401 is because it has been well studied. We use the SET-IV$_b$ dataset of MHCBench database according to previous work[26]. The ratio of positive/negative examples is 1:1, and redundant duplicate peptides have been removed. A set of 584 peptides are obtained.

Three other datasets for the HLA class alleles HLA-DRB1*0101, HLA-DRB1*0301 and HLA-DRB1*1101 are obtained from the MHCBN database[33]. The original dataset for HLA-DRB1*0101 consists of 585 binder and 131 non-binder peptides. The dataset for HLA-DRB1*0301 consists of 219 binder and 154 non-binder peptides. The dataset for HLA-DRB1*1101 consists of 278 binder and 96 non-binders. There are quite unbalanced number of binders and non-binders in these datasets, therefore we randomly choose binders to keep the ratio of positive/negative examples being 1:1. The length of most peptides is longer than 12 and the average length of HLA-DRB1*0401 is 26.34, much longer than the other alleles drawn from MHCBN.The size of original datasets is listed in **Table 2.**

Table 2. Size of peptides in original datasets for 4 HLA-class II molecules

HLA allele	Binder size	Non-binder	Length			Database
			Min	Max	Ave.	
DRB1*0401	292	292	19	35	26.34	MHCBench
DRB1*0101	131	131	8	25	15.44	MHCBN
DRB1*0301	154	154	8	24	17.80	MHCBN
DRB1*1101	96	96	8	24	14.39	MHCBN

3.2 Testing of the Benchmark Data for HLA-DRB1*0401 Allele

The HLA-DRB1*0401dataset is randomly partitioned into five parts. Each time four out of the five parts are used for training, while the remainder part is used for evaluation. The average performance of 5-fold cross validation is taken as the final result. The following measurements are used in our approach:

- Sensitivity: the proportion of correctly predicted binders to all predicted binders, TP/(TP+FN)
- Specificity: the proportion of correctly predicted non-binders to all predicted non-binders, TN/(TN+FP)
- PPV: the probability that a predicted binder will actually be a binder, TP/(TP+FP)
- NPV: the probability that a predicted non-binder will be a real non-binder, TN/(TN+FN)
- Accuracy: the percent of correct predictions
- MCC: Matthews correlation coefficient, which is defined as follows:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FN)(TP+FP)(TN+FP)(TN+FN)}} \tag{8}$$

50 different models are generated and the best performance is achieved when n=2 and λ=0.72. The results of our approach on the HLA-DRB1*0401 dataset are compared with those obtained from the Motif-, Matrix- and ANN-based methods. Since all methods were evaluated on the same dataset, they are directly comparable. **Table 3** shows the comparative results. Note that the results from other methods except string kernel (see our previous work) are directly reproduced from MHCBench.

It is observed that among the proposed methods, our approach has the best MCC than other methods. It is also observed that some motif-based methods can get high specificity (>82%) but very low sensitivity (<35%).

Table 3. The performance of different prediction algorithms on MHCBench's SET-IVb dataset for HLA-DRB1*0401

Method	Ref.	Sen (%)	Spec (%)	PPV (%)	NPV (%)	Acc (%)	MCC
Motif	Max, 1993[35]	51.03	67.12	60.82	57.82	59.08	0.1839
	Chicz, 1993[4]	34.85	**82.95**	**77.84**	42.55	52.54	0.1906
	Rammen-nse, 1995[9]	68.15	71.92	70.82	69.31	70.03	0.4010
Matrix	Marshal, 1995[36]	68.15	68.84	68.62	68.37	68.49	0.3699
	Brusic, 1998[6]	63.70	64.04	63.92	63.82	63.87	0.2774
	Borras-Cuesta, 2000[37]	59.59	57.53	58.39	58.74	58.56	0.1713
ANN	Brusic, 1998 [6]	64.04	64.38	64.26	64.16	64.21	0.2842
String kernel	Yu, 2006[26]	72.48	70.12	70.63	72.05	71.40	0.4265
Our approach		**74.83**	72.52	73.46	**73.81**	**73.63**	**0.4731**

One reason for the poor performance of some methods is that they relied heavily on the training dataset and they did not use non-binding data. Another reason is that string kernel is more suitable for classifying binders and non-binders. A notable observation is that the performance of ANN-based method appears to be very poor because it requires that the core region in this dataset has to be determined prior to the prediction. It is difficult to extract proper 9mer core region from peptides whose average length is 26.

Some recent methods are not compared on MHCBench dataset because they focused on different HLA class II molecules[3] or used unpublished data. Burden[1] presented predictive Bayesian neural network models, and only processed 9AA peptides. Bhasin[38] also used a SVM method with RBF kernel on another dataset and they declared to achieve an accuracy of 86.1%, but the performance on a blind dataset is yet to be experimentally validated.

3.3 Evaluation for Other HLA Class II Alleles

Three datasets for HLA class II alleles HLADRB1*0101, HLA-DRB1*0301 and HLA-DRB1*1101 are obtained from the MHCBN database as described in **Table 2**.

Our approach is evaluated with 5-folder cross validation on the three datasets. The results are list in **Table 4.** The corresponding λ parameter is 0.6, 0.3, and 0.6, respectively. The length of sub-strings *n* is 2.

Table 4. The prediction result for other MHC class II alleles

HLA allele	Sen (%)	Spec (%)	PPV (%)	NPV (%)	Acc (%)	MCC
DRB1*0101	77.84	82.04	78.50	77.89	79.02	0.5808
DRB1*0301	75.99	81.17	79.29	77.51	78.58	0.5697
DRB1*1101	79.89	84.68	84.66	79.78	82.64	0.6450

The result is also compared against those obtained from TEPITOPE (ProPred) [39] and LP(iterative learning model) [24]. The performance is now evaluated by the area under the ROC curve (A_{ROC}). The ROC curve uses the combination of specificity and sensitivity to measure the performance. In a ROC curve, the A_{ROC} value have a range of [0, 1]. A purely random prediction model would have an A_{ROC} of 0.5, and the closer A_{ROC} is to 1, the better the performance is. Our results are shown in **Table 5** and the ROC curve in **Fig 1**.

Table 5. The average A_{ROC} values for three different methods

Method	DRB1*0101	DRB1*0301	DRB1*1101
ProPred	0.842	0.585	**
LP	0.779	0.721	**
our approach	**0.869**	**0.802**	**0.857**

** These methods had no results for HLA-DRB1*1101 in [24].

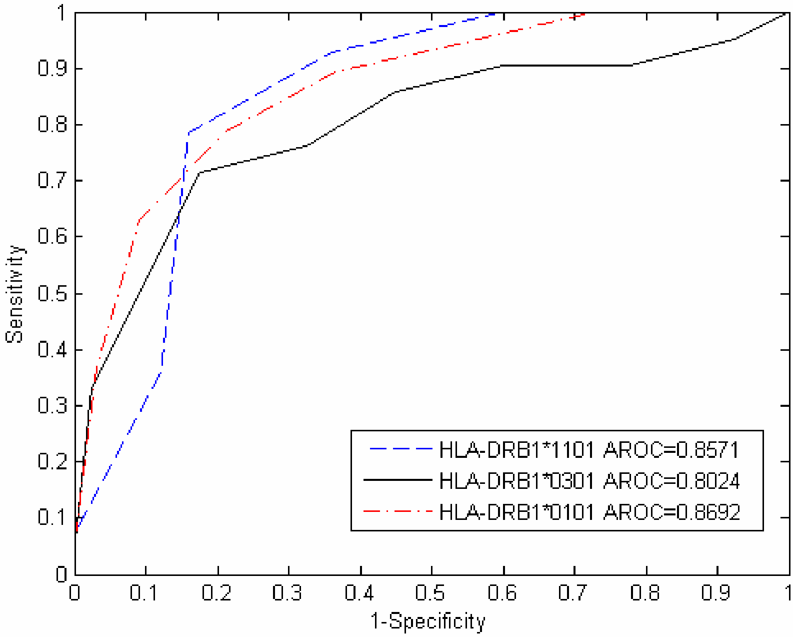


Fig. 1. Roc curves of three HLA class II alleles predicted by our approach

The experiment shows that our approach contributes promising results for the prediction of other HLA class II binding peptides. Note that the results in **Table 4** are significantly better than those shown in **Table 3**. This is because the datasets used in the two experiments are different (peptides in the first experiment are significantly longer than those in the second experiment, **see Table 2**). Another difference is that the data in the first experiment is larger scale and all the binding peptides have been experimentally verified. Moreover, the peptides used in the first experiment are natural peptides. It is observed that predicting longer peptides is more difficult.

4 Discussion and Conclusion

Predicting the binding between HLA molecules and short peptides is still a major task for immunoinformatics, particularly for HLA class II alleles due to the various lengths of reported binding peptides and undetermined core regions.

Most previous methods impose a variety of uncertain biological assumptions, for example, assuming that a fixed length of continuous amino acids (usually 9mers) stand for the binding region. A number of methods have been proposed to extract core regions. However, not all HLA II binding peptides has such a region or the region may be discontinuous.

In this paper, we demonstrated a novel kernel-based approach for predicting binding peptides for HLA II alleles, which integrates BLOSUM matrix with string kernel. The approach is competitive to handle various lengths of peptides (thus no need to extract core regions) and also exhibits better results than other methods. The approach is simple, easy to implement and do not impose many biological assumptions. In our approach, more biological knowledge and meaning is exploited by applying substitution scores in BLOSUM matrix. This is the reason why our approach outperforms others in our belief.

Acknowledgement

The work was supported by Chinese Natural Science Foundation under grant No.60272019 and 60321002.

References

1. Burden, F.R. and D.A. Winkler, *Predictive Bayesian neural network models of MHC class II peptide binding.* Journal of Molecular Graphics and Modelling, 2005. **23**: p. 481-489.
2. Bhasin, M. and G.P.S. Raghava, *SVM based method for predicting HLA-DRB1(*)0401 binding peptides in an antigen sequence.* Bioinformatics, 2004. **20**(3): p. 421-423.
3. Bozic, I., G.L. Zhang, and V. Brusic, *Predictive vaccinology: Optimisation of predictions using support vector machine classifiers*, in *Intelligent Data Engineering And Automated Learning Ideal 2005, Proceedings*. 2005. p. 375-381.
4. Chicz, R.M., et al., *Specificity and promiscuity among naturally processed peptides bound to HLA-DR alleles.* J. Exp. Med., 1993. **178**(1): p. 27-47.

5. Chen, Y., et al., *Naturally processed peptides longer than nine amino acid residues bind to the class I MHC molecule HLA-A2.1 with high affinity and in different conformations.* The journal of immunology, 1994. **152**: p. 2874-2881.
6. Brusic, V., et al., *Prediction of MHC class II-binding peptides using an evolutionary algorithm and artificial neural network.* Bioinformatics, 1998. **14**: p. 121-130.
7. Schueler-Furman, O., et al., *bound peptides: a study of 23 complexes.* Folding design, 1998. **3**: p. 549-564.
8. Froloff, N., A. Windemuth, and B. Honig, *On the calculation of binding free energies using continuum methods: Application to MHC class I protein-peptide interactions.* Protein Sci, 1997. **6**(6): p. 1293-1301.
9. Rammensee, H.G., T. Friede, and S. Stevanovic, *Mch Ligands And Peptide Motifs - First Listing.* Immunogenetics, 1995. **41**(4): p. 178-228.
10. Meister, G.E., et al., *2 Novel T-Cell Epitope Prediction Algorithms Based On Mhc-Binding Motifs - Comparison Of Predicted And Published Epitopes From Mycobacterium-Tuberculosis And Hiv Protein Sequences.* Vaccine, 1995. **13**(6): p. 581-591.
11. K Falk, O Roetzschke, S Stevanovie, G Jung, HG Rammensee., *Allele-specific motifs revealed by sequencing of self-peptides eluted from MHC molecules*, in *Nature*. 1991. p. 290-296.
12. Hammer, J., et al., *New Methods To Predict Mhc-Binding Sequences Within Protein Antigens.* Current Opinion In Immunology, 1995. **7**(2): p. 263-269.
13. Hammer, J., et al., *Precise Prediction Of Major Histocompatibility Complex Class-Ii Peptide Interaction Based On Peptide Side-Chain Scanning.* Journal Of Experimental Medicine, 1994. **180**(6): p. 2353-2358.
14. Mallios, R.R., *Class II MHC quantitative binding motifs derived from a large molecular database with a versatile iterative stepwise discriminant analysis meta- algorithm.* Bioinformatics, 1999. **15**(6): p. 432-439.
15. Sturniolo, T., et al., *Generation of tissue-specific and promiscuous HLA ligand databases using DNA microarrays and virtual HLA class II matrices.* Nature Biotechnology, 1999. **17**(6): p. 555-561.
16. Rammensee, H.G., et al., *SYFPEITHI: database for MHC ligands and peptide motifs.* Immunogenetics, 1999. **50**(3 - 4): p. 213-219.
17. Mamitsuka, H.M., *Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models.* Proteins, 1998. **33**: p. 460-474.
18. Donnes, P. and A. Elofsson, *Prediction of MHC class I binding peptides, using SVMHC.* Bmc Bioinformatics, 2002. **3**(1):p.25-32.
19. Zhao, Y.D., et al., *Application of support vector machines for T-cell epitopes prediction.* Bioinformatics, 2003. **19**(15): p. 1978-1984.
20. Wan, J., et al., *SVRMHC prediction server for MHC-binding peptides.* BMC Bioinformatics, 2006. **7**(1): 463.
21. Brusic, V., et al. *Data Cleansing for Computer Models: A Case Study from Immunology.* in *Proceedings of ICONIP99, The Sixth International Conference on Neural Information Processing*. 1999.
22. Nielsen, M., et al., *Improved prediction of MHC class I and class II epitopes using a novel Gibbs sampling approach.* Bioinformatics, 2004. **20**(9): p. 1388-1397.
23. Karpenko, O., J. Shi, and Y. Dai, *Prediction of MHC class II binders using the ant colony search strategy*. Vol. 35. 2005: Elsevier Science. 147-156.
24. Murugan, N. and Y. Dai, *Prediction of MHC class II binding peptides based on an iterative learning model.* Immunome Research, 2005. **1**(1): 6.

25. Doytchinova, I.A. and D.R. Flower, *Towards the in silico identification of class II restricted T-cell epitopes: a partial least squares iterative self-consistent algorithm for affinity prediction.* Bioinformatics, 2003. **19**(17): p. 2263-2270.
26. Yu, H., X. Zhu, and M. Huang. *Using String Kernel to Predict Binding Peptides for MHC Molecules*. in *the 8th International Conference on Signal Processing*. 2006. Guilin, China: IEEE Press. p. 2499-2502.
27. Saigo, H., et al., *Protein homology detection using string alignment kernels.* Bioinformatics, 2004. **20**(11): p. 1682-1689.
28. Zhao, X.M., Y.M. Cheung, and D.S. Huang, *A novel approach to extracting features from motif content and protein composition for protein sequence classification.* Neural Networks, 2005. **18**(8): p. 1019-1028.
29. J, M., *Functions of positive and negative type and their connection with the theory of integral equations.* Philos. Trans. Soc, 1909. **209**: p. 415-446.
30. Lodhi, H., et al., *Text classification using string kernels.* Journal Of Machine Learning Research, 2002. **2**(3): p. 419-444.
31. Brown, M.P.S., et al., *Knowledge-based analysis of microarray gene expression data by using support vector machines.* PNAS, 2000. **97**(1): p. 262-267.
32. Brusic, V., G. Rudy, and L.C. Harrison, *MHCPEP, a database of MHC-binding peptides: update 1997.* Nucl. Acids Res., 1998. **26**(1): p. 368-371.
33. Bhasin, M., H. Singh, and G.P.S. Raghava, *MHCBN: a comprehensive database of MHC binding and non-binding peptides.* Bioinformatics, 2003. **19**(5): p. 665-666.
34. Joachims, T., *Making large-Scale SVM Learning Practical.* Advances in Kernel Methods - Support Vector Learning, ed. B.S.a.C.B.a.A. Smola. 1999: MIT-Press.
35. Max, H., et al., *Characterization of peptides bound to extracellular and intracellular HLA-DR1 molecules*, in *Human immunology*. 1993, Elsevier/North-Holland.: [New York, N.Y.]. p. 193.
36. Marshall, K.W., et al., *Prediction of peptide affinity to HLA DRB1*0401.* J Immunol, 1995. **154**(11): p. 5927-5933.
37. Borras-Cuesta, F., et al., *Specific and general HLA-DR binding motifs: comparison of algorithms.* Human Immunology, 2000. **61**(3): 266.
38. Bhasin, M. and G.P.S. Raghava, *Prediction of CTL epitopes using QM, SVM and ANN techniques.* Vaccine, 2004. **22**(23-24): p. 3195-3204.
39. Singh, H. and G.P.S. Raghava, *ProPred: prediction of HLA-DR binding sites.* Bioinformatics, 2001. **17**(12): p. 1236-1237.

A Database for Prediction of Unique Peptide Motifs as Linear Epitopes

Margaret Dah-Tsyr Chang[1], Hao-Teng Chang[1], Rong-Yuan Huang[1],
Wen-Shyong Tzou[2], Chih-Hong Liu[3], Wei-Jun Zhung[3], Hsien-Wei Wang[3],
Chun-Tien Chang[4], and Tun-Wen Pai[3,*]

[1] Institute of Molecular and Cellular Biology & Department of Life Science,
National Tsing Hua University, Hsinchu, Taiwan, Republic of China
[2] Institute of Bioscience and BioTechnology, National Taiwan Ocean University,
Keelung, Taiwan, Republic of China
[3] Department of Computer Science and Engineering, National Taiwan Ocean University,
Keelung, Taiwan, Republic of China
[4] Department of Computer Science, National Tsing Hua University, Hsinchu, Taiwan,
Republic of China
twp@mail.ntou.edu.tw

Abstract. A linear epitope prediction database (LEPD) is designed for identification of unique peptide motifs (UPMs) as specific linear epitopes for all protein families defined by Pfam. The UPMs in LEPD are extracted from each protein family by employing reinforced merging techniques that merge the primary unique patterns into a consecutive peptide based on the neighboring relationships and various levels of parameter settings. These merged peptide motifs are examined using the physicochemical and structural propensity scales for antigenic characteristics and are verified by employing background model analysis for specificity. The filtered UPMs with high antigenicity and specificity are considered as linear epitopes that provide important information for designing antibodies and vaccines. The predicted epitopes of each protein family in the LEPD can be searched in a straightforward manner, and the corresponding chemical properties be displayed in graphical and tabular formats. To verify the specificity of the predicted epitopes, each identified UPM is analyzed by scanning over the complete genomes of a series of model organisms. For any query protein possessing a resolved 3D structure, the proposed database also provides interactive visualization of the protein structures for allocation and comparison of the predicted linear epitopes. The accuracy of the prediction algorithm is evaluated to be higher than 70% in terms of mapping a UPM as a linear epitope as compared to the known databases.

Keywords: unique peptide motif, linear epitope, antigenicity, specificity, protein family.

Availability: The linear epitope prediction database is available at http://140.121.196.30/LEPD, and the details of the proposed algorithm can be referred to the documents as described previously (1,2).

[*] Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 430–440, 2007.

1 Introduction

Antigen-antibody interactions facilitate the identification and characterization of specific molecular recognition *in vivo* and *in vitro*. In general, the most remarkable features of this special class of protein-protein interactions are the high affinity and strict specificity among antibodies and their respective antigens. Structural determination by employing NMR and X-ray crystallography have revealed that antibodies recognize the unique conformations and spatial locations composed of sequential or gapped regions on the antigen surface (3,4). Therefore, epitopes are defined as the portions of antigen molecules exhibiting complementary structural interactions with the antigen determinants of antibodies (5,6). B-cell epitopes are classified as either linear or discontinuous epitopes (conformational epitopes). The former comprise a single continuous stretch of amino acids within a protein sequence, while the latter possess distantly separated fragments in a protein sequence resulting in physical proximity due to the protein folding mechanism. Generally speaking, there exist three types of computer-based methods for predicting potential B-cell epitopes. The first method applies physicochemical and structural propensity scales for linear epitopes from the primary sequence of a protein. The second method is based on a comparative mechanism that takes into consideration the amino acid sequence and the structural similarity among proteins in terms of known antigenicities. The last method is derived from the integration of structural features such as surface accessibility and loop characteristics with an existing B-cell epitope motif database (7). Most available epitope prediction tools are designed to analyze a single query protein sequence at a time (7,8,9,10,11). These tools mainly focus on analyzing the chemical property of the specified protein sequence itself or comparing it directly with the known epitope databases. In this study, unlike the conventional methods, we propose a pre-analysis module that performs multiple sequence-based searches followed by a comparison operation to extract potential epitopes from a set of closely related proteins and generates a novel database for linear epitopes based on their respective antigentic characteristics. We believe that the predicted epitopes obtained by comparing with their related family sequences can provide more accurate results from the viewpoint of designing protein-specific antibodies.

A protein family, and its related domains, is defined as a set of proteins that possess a common evolutionary origin. The relationship between the common biophysical and biochemical properties of a protein family can be easily observed from the high sequence identity, structural homology, or similar biological functions (12). However, certain multifunctional enzymes possess high sequence identity but differential biological functions other than the common catalytic abilities; this is presumably due to the molecular interactions with different cellular compartments. Therefore, the identification of the localizations and compositions of the unique peptide motifs (UPM) in each member of a protein family to establish correlation with their unique functions could yield useful information. In the proposed database, the categorized protein families were defined by Pfam—a comprehensive database containing 8957 families in the current release version (21.0). Each family is manually curated and is represented by two multiple sequence alignments, two profile hidden Markov models (HMMs), and an annotation file (13).

To evaluate the physicochemical properties of the query sequences, approximately 500 normalized amino acid scales were identified from the AAindex database (14); each scale assigns a value to each of the 20 amino acids. To enhance the antigenicity of the searched unique peptide motifs, the proposed LEPD also provides evaluation functions for the users. Various combinations of parameters and weighting coefficients can be selected to visualize the positions of the linear epitopes.

2 Algorithms for Linear Epitope Prediction

Linear epitopes are predicted by bottom-up merging techniques that identify the UPMs from all classified protein families defined by Pfam (1,13). The proposed system employs a sequence-based search algorithm that consists of a three-phase operation comprising clustering, searching, and merging modules. The clustering phase based on hierarchical clustering techniques is an optional process that facilitates the classification of the 20 amino acids according to the specified BLOSUM/PAM series of matrices (15,16). The clustered groups can also be simply customized according to the user's preference. In the search phase, the system executes exact or approximate string matching to extract the fundamental UPMs referred to as primary patterns. The length of the primary pattern is selected as a parameter for extracting the appropriate fundamental units in order to construct a representative UPM. The rule of thumb for primary pattern lengths is that a longer length is set for similar sequences and a shorter one for dissimilar sequences. In other words, if the pattern length selected as the primary pattern in a diverging sequence set is longer than required, then a larger set of fundamental unique patterns is obtained resulting in a high false-positive rate. Therefore, a suitable primary pattern length should be selected so as not to produce several fundamental UPMs, and the identified patterns are expected to merge to form a longer merged segment. To calculate the optimal length of the primary patterns and formulate a linear epitope database, we propose an intelligent decision algorithm in this study. The general rule for determining the optimal primary pattern length is to find the maximal production between the percentage of unique primary patterns and the percentage of non-merged sites. The percentage of unique primary patterns is calculated as the ratio of the number of unique primary patterns with a specified length to the total number of all possible segments with the same specified length. The percentage of non-merged sites is computed as the ratio of the number of non-merged sites after the merging operations with the specified criteria to the total number of residues of all the imported sequences. In the proposed system, we investigate the pattern lengths in the range of 8 to 15 as the optimal length for the identification of linear epitopes. After the length setting for the primary pattern is determined, the search module executes the Boyer-Moore algorithm to efficiently retrieve all primary patterns based on previous clustering definitions. Each searched fundamental segment is analyzed based on its appearance frequencies, and its representation level of uniqueness is calculated for the subsequent merging processes.

In the last phase of the merging algorithms, the system initiates a methodology to extract unique primary patterns by the bottom-up merging processes. The construction of a merged segment *w* from overlapped primary patterns *u* and *v* is formulated in Eq. (1).

$$w(i) = \begin{cases} u(i) & if\ 1 \le i \le m; \\ v(i-m+l) & if\ m+1 \le i \le 2m-l, \end{cases} \tag{1}$$

where $w(i)$ is the i^{th} residue of *w* ; *m*, the length of *u* and *v* ; and *l*, the number of overlapped residues. In addition, as the merged UPMs from the query sequences are allocated, the LEPD ranks the identified segments according to the four default key features to represent the following antigenic properties of amino acids: hydrophilicity, charge, number of prolines, and the proximity of the segments towards the *N*- or *C*-terminal end of the protein. A higher rank indicates stronger antigenicity in the UPMs. Consequently, segments with a higher rank may be suggested to serve as a suitable linear epitope or peptide antigen for generation of a specific antibody (1). However, the coarse definition of the antigenicity of distinct amino acids may not be sufficient to represent the details of the chemical propensity of each amino acid. Therefore, the proposed system also provides the well-known physicochemical and structural propensity scales for the user's selection. Details regarding this are discussed in the subsequent section.

3 Physico-Chemical and Structural Propensity Scales

When the UPMs extracted from a protein family are ordered, the antigenicity of each candidate can be reexamined by using the physicochemical and structural propensity scales. Herein, several properties are considered for recognizing a UPM as an ideal epitope. They include hydrophilicity/hydrophobicity, bent features (secondary structure, mainly beta-turns), surface accessibility, polarity, and mobility and/or flexibility characteristics. These basic biophysical parameters are calculated with various weights and considered for the determination of potential epitopes. In order to extract the local optimal and continuous segments from a sequence, we consider the profile of a chemical propensity scale as a signal function $f(i) = \sum_j c_j f_j(i)$, where *i* is the i^{th} residue, $f_j(i)$ represents the scores from the j^{th} chemical property at the i^{th} residue, and c_j is the weighting coefficient of the j^{th} chemical property. Using the derived chemical propensity function, we can design an algorithm to extract continuous segments w_f with antigenicity higher than a defined threshold value. To efficiently extract a continuous stretch from a sequence based on its related antigentic profiles, we employ the techniques of grayscale mathematical morphology to filter out the segments with local maximal scales. Four basic morphological operations, including dilation, erosion, opening, and closing operations, are defined in theory. A function of the chemical propensity profile, *f*(*i*), is represented as a discrete signal in a real-valued function on ***R***. The signal has values only at the discrete point, where *i* assumes only integer values, and is represented at the i^{th} residue of the query sequence. Hence, the basic grayscale morphological operations can be denoted as follows:

$$\text{Erosion: } \mathcal{E}(f,g) = f\Theta g = \min_{x\in g,\forall i} \{f(i+x)\}$$

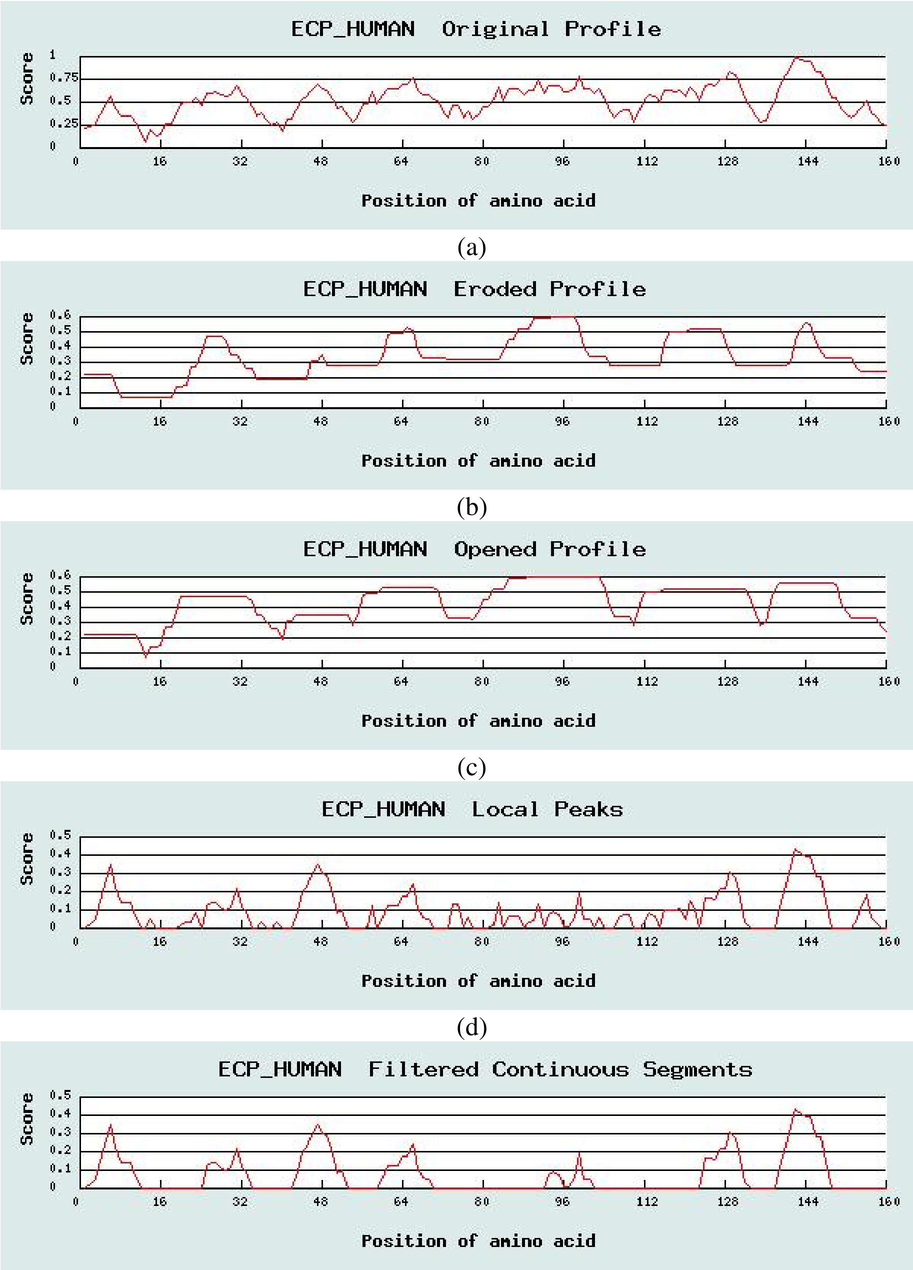
$$\text{Dilation: } \mathcal{D}(f,g) = f\oplus g = \max_{x\in g,\forall i} \{f(i+x)\}$$

$$\text{Opening: } \mathcal{O}(f,g) = f\circ g = \mathcal{D}(\mathcal{E}(f,g),g) = [f\Theta g]\oplus g$$

$$\text{Closing: } \mathcal{C}(f,g) = f\bullet g = \mathcal{E}(\mathcal{D}(f,g),g) = [f\oplus g]\Theta g$$

where *i* and *x* denote the position of an amino acid under evaluation, *f* is a real-valued discrete signal, and *g* is a predetermined function of the structuring element. The dilation of *f* by *g* at the i^{th} position is the maximal value of all points *f*(*i* + *x*) such that $x\in g$, and the erosion of *f* by *g* at the i^{th} position is the minimal value of all points *f*(*i* + *x*) such that $x\in g$. The closing operation involves first considering the dilation of *f* by *g* followed by an erosion operation by *g*. For the opening operation, the transformation is considered to be the erosion of *f* by *g* prior to performing the dilation operation on the eroded function by using *g*.

Both the opening and closing operations use function *g* once as a structuring element to compensate for the shift effect due to the first erosion or dilation when *g* is not a symmetric function. Grayscale morphological dilation and erosion are defined from a functional viewpoint where the dilation and erosion operations cause the “expanding” and “shrinking,” respectively, of the envelope of a function. The opening transformation cuts off the local positive peaks, whereas the closing transformation fills up the local valleys from the original function. Therefore, in this study, we employ morphological opening operations to first cut off the local peaks, and then the difference between the original and opened function are obtained to reveal the local peaks. Once the local peaks are derived, we only need to evaluate the length of the continuous segments. The lengths of the corresponding structuring elements are selected according to the required epitope length. If we assume the typical length of a peptide segment for an epitope to be between 5 and 15, then the size of the structuring function can be set as 5. We present an example of local peak extraction from an original chemical propensity profile. In Figure 1(a), the original chemical propensity profile of the ECP_HUMAN sequence is calculated and normalized in the range of [0,1] by considering the following default weighted coefficients: 0.4 for secondary structure feature, 0.3 for hydrophilicity property, 0.15 for surface accessibility, and 0.15 for flexibility. The eroded and opened profiles are shown in Figure 1(b) and 1(c), respectively. The difference between the original and opened functions is obtained and displayed in Figure 1(d). Finally, Figure 1(e) demonstrates the application of a size filter to extract the continuous segments that satisfy the minimal length requirement.



(a)

(b)

(c)

(d)

(e)

Fig. 1. Chemical propensity profiles of the ECP_HUMAN sequence and the corresponding local continuous segment extraction by the combination of morphological operations

4 Results and Discussion

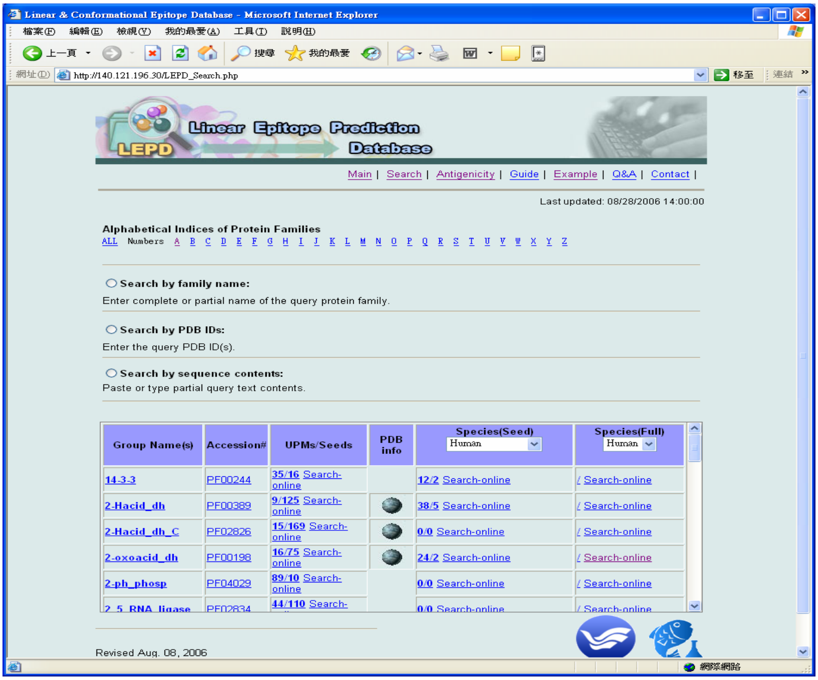
The LEPD provides complete identified UPMs with various antigenic priorities from all the clustered protein family sequences listed in Pfam. Moreover, the R_EMUS system (http://140.121.196.30/REMUS/) facilitates the online analysis of linear epitope prediction. Using this system, the users can input any query protein data set for analysis rather than selecting the default protein family defined by Pfam. Both systems are implemented on IIS with Windows Server as the operating system and the web interface is written in PHP.

To illustrate the practical application of the LEPD, the major operation steps are demonstrated in the following figures. Figure 2 shows the keyword search interface for a preferred protein family and the corresponding identified linear epitopes. In the current version, 8957 protein families available in Pfam (release 21.0) have been analyzed completely. The UPMs of each query protein family are arranged alphabetically by the interface. As shown in Figure 3, a user can click on a specific protein family displayed on the screen shown in Figure 2; on clicking, the predicted linear epitopes are identified and listed according to their antigenicity rank.

In Figure 4, the selected chemical properties, including hydrophilicity, hydrophobicity, surface accessibility, polarity, secondary structure, etc., for each predicted linear epitope are shown for advanced analyses. Moreover, the resolved 3D structural information of all protein families or domains can be displayed and analyzed simultaneously for their secondary structural features, surface accessibilities, and relative geometrical positions. The predicted linear epitopes can be displayed in an interactive manner in a web browser for 3D visualization by incorporating with the Jmol (http://jmol.sourceforge.net/) technology, as shown in Figure 5.

The linear epitopes can be validated by a variety of biochemical and biophysical methods such as Western blotting, enzyme-linked immunosorbent assay (ELISA), immunoprecipitation, and X-ray crystallography. The prediction results can provide researchers valuable information regarding peptide antigen design and the structural determination of the heterocomplexes formed by specific antigen-antibody interactions. In this study, the performance of the bottom-up merging algorithms for the prediction of linear epitopes is evaluated by adopting the epitopes of human monoclonal antibodies retrieved from the website of Santa Cruz Biotechnology, Inc. (http://www.scbt.com/), which focuses on the ongoing development of antibodies for research. Among the 21337 entries listed in the database, 3450 monoclonal antibodies with a specified epitope length of less than 300 amino acid residues are generated against human antigens; these are classified into 70 human protein families containing a total of 260 protein sequences. Each set of the protein family sequences is collected from seed sequences of Pfam database and analyzed employing the bottom-up merging algorithms. It is found that 582 UPMs are located within the antibody-antigen recognition sites of 55 monoclonal antibodies; this indicates that the average accuracy of matching at least one of the UPMs with the reported epitopes of the antibodies of selected protein family is 78.6% (55/70). It should be noted that Pfam database is constructed based on sequence homology of conservative functional domains, therefore the most variable segments as potential linear epitopes of protein family sequences, frequently the terminal regions, are excluded in the database. As compared with our previous reports in which full length protein family sequences were derived from GeneBank database (1,2), less UPMs were obtained in LEPD.

To further evaluate the specificity of the searched UPMs, we conduct background model analysis by scanning the identified UPMs over the completed genome species. Since all the UPMs listed in our database are derived by identifying the common sequence stretches during the multiple sequence comparison of closely related protein families, it is expected that the identified UPMs are quite rare in the rest areas of the genomes. Considering the RNase A-type families as an example, we obtain 30 seed sequences clustered in the family set and 47 UPMs are allocated under the default settings. As expected, when each predicted epitope is scanned over the completed genome provided by the NCBI database (ftp://ftp.ncbi.nih.gov/genomes/), only 6 out of the 47 candidates appear in more than one species. This advanced analysis provides an additional filter for improving the specificity of antigenic epitope selection. In summary, we have developed a novel database for the prediction of the linear epitopes located in all protein families listed in Pfam. Antigens designed based on the identified UPMs are advantageous in differentiating one member of a large protein family due to the nature of specific molecular recognition.


Linear & Conformational Epitope Database - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
上一頁
搜尋
我的最愛
網址(D)
http://140.121.196.30/LEPD_Search.php
移至
連結
LEPD
Linear Epitope Prediction Database
網際網路

Main | Search | Antigenicity | Guide | Example | Q&A | Contact |

Last updated: 08/28/2006 14:00:00

Alphabetical Indices of Protein Families

ALL Numbers A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

○ Search by family name:

Enter complete or partial name of the query protein family.

○ Search by PDB IDs:

Enter the query PDB ID(s).

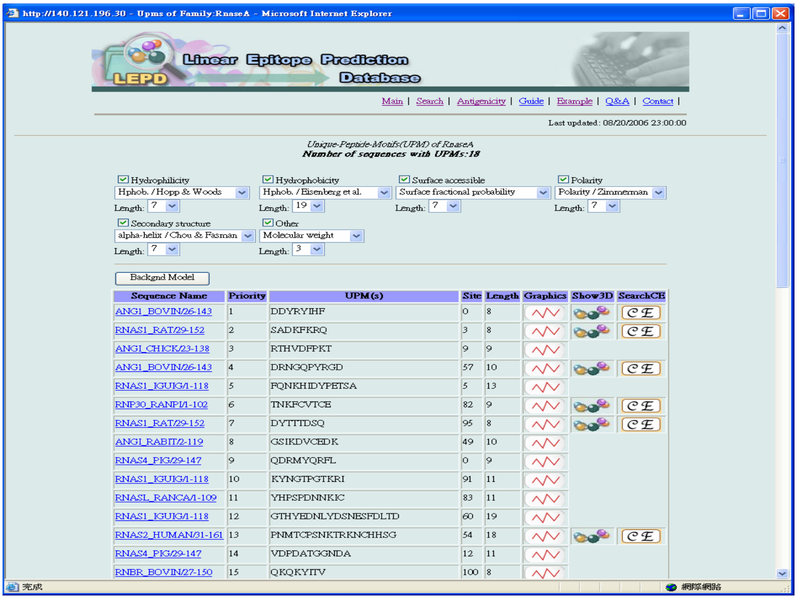
○ Search by sequence contents:

Paste or type partial query text contents.

Group Name(s)	Accession#	UPMs/Seeds	PDB info	Species(Seed) Human	Species(Full) Human
14-3-3	PF00244	35/16 Search-online		12/2 Search-online	/ Search-online
2-Hacid_dh	PF00389	9/125 Search-online	●	38/5 Search-online	/ Search-online
2-Hacid_dh_C	PF02826	15/169 Search-online	●	0/0 Search-online	/ Search-online
2-oxoacid_dh	PF00198	16/75 Search-online	●	24/2 Search-online	/ Search-online
2-ph_phosp	PF04029	89/10 Search-online		0/0 Search-online	/ Search-online
2_5_RNA_ligase	PF02834	44/110 Search-		0/0 Search-online	/ Search-online

Revised Aug. 08, 2006

Fig. 2. Search interface and the related linear epitopes of each protein family listed in Pfam


http://140.121.196.30 - Upms of Family:RnaseA - Microsoft Internet Explorer
LEPD
Linear Epitope Prediction Database
Main | Search | Antigenicity | Guide | Example | Q&A | Contact |
完成
網際網路

Last updated: 08/20/2006 23:00:00

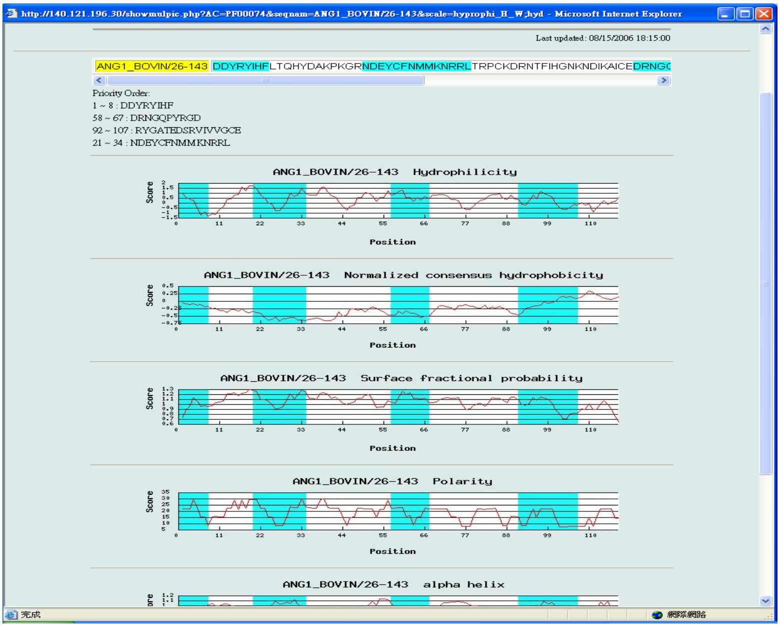
Unique-Peptide-Motifs(UPM) of RnaseA
Number of sequences with UPMs:18

☑ Hydrophilicity: Hphob. / Hopp & Woods, Length: 7
☑ Hydrophobicity: Hphob. / Eisenberg et al., Length: 19
☑ Surface accessible: Surface fractional probability, Length: 7
☑ Polarity: Polarity / Zimmerman, Length: 7
☑ Secondary structure: alpha-helix / Chou & Fasman, Length: 7
☑ Other: Molecular weight, Length: 3

Backgnd Model

Sequence Name	Priority	UPM(s)	Site	Length	Graphics	Show3D	SearchCE
ANG1_BOVIN/26-143	1	DDYRYIHF	0	8	∧/∨	●	CE
RNAS1_RAT/29-152	2	SADKFKRQ	3	8	∧/∨	●	CE
ANGI_CHICK/23-138	3	RTHVDFPKT	9	9	∧/∨		
ANG1_BOVIN/26-143	4	DRNGQPYRGD	57	10	∧/∨	●	CE
RNAS1_IGUIG/1-118	5	FQNKHIDYPETSA	5	13	∧/∨		
RNP30_RANPI/1-102	6	TNKFCVTCE	82	9	∧/∨	●	CE
RNAS1_RAT/29-152	7	DYTTTDSQ	95	8	∧/∨	●	CE
ANGI_RABIT/2-119	8	GSIKDVCEDK	49	10	∧/∨		
RNAS4_PIG/29-147	9	QDRMYQRFL	0	9	∧/∨		
RNAS1_IGUIG/1-118	10	KYNGTPGTKRI	91	11	∧/∨		
RNASL_RANCA/1-109	11	YHPSPDNNKIC	83	11	∧/∨		
RNAS1_IGUIG/1-118	12	GTHYEDNLYDSNESFDLTD	60	19	∧/∨		
RNAS2_HUMAN/31-161	13	PNMTCPSNKTRKNCHHSG	54	18	∧/∨	●	CE
RNAS4_PIG/29-147	14	VDPDATGGNDA	12	11	∧/∨		
RNBR_BOVIN/27-150	15	QKQKYITV	100	8	∧/∨		

Fig. 3. Specified RNase A-type protein family and the corresponding linear epitopes in the decreasing order of antigenticity

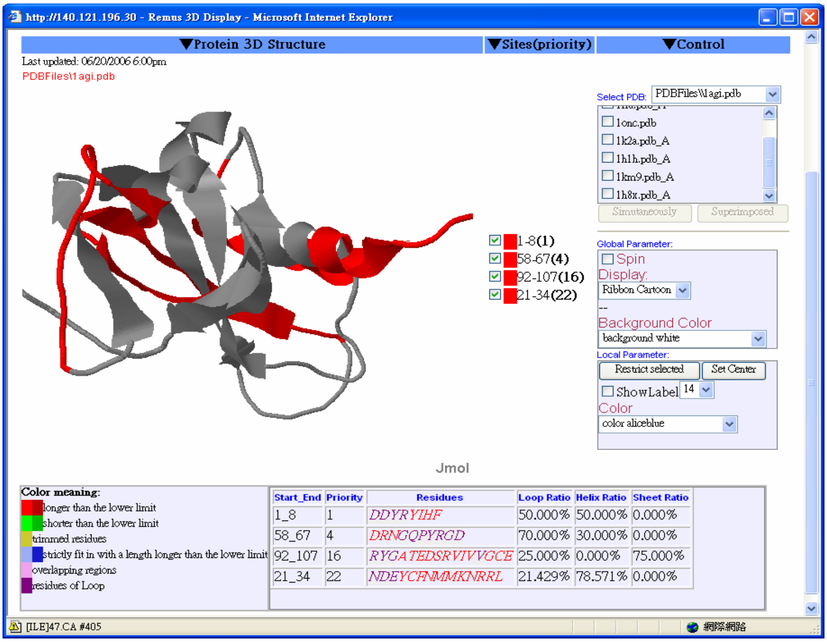

http://140.121.196.30/showmulpic.php?AC=PF00074&seqnam=ANG1_BOVIN/26-143&scale=hyprophi_H_W;hyd - Microsoft Internet Explorer
ANG1_BOVIN/26-143 Hydrophilicity
Score
2
1.5
1
0.5
0
−0.5
−1
−1.5
0
11
22
33
44
55
66
77
88
99
110
Position
ANG1_BOVIN/26-143 Normalized consensus hydrophobicity
Score
0.5
0.25
0
−0.25
−0.5
−0.75
Position
ANG1_BOVIN/26-143 Surface fractional probability
Score
1.3
1.2
1.1
1
0.9
0.8
0.7
0.6
Position
ANG1_BOVIN/26-143 Polarity
Score
35
30
25
20
15
10
5
Position
ANG1_BOVIN/26-143 alpha helix
1.2
1.1
完成
網際網路

Last updated: 08/15/2006 18:15:00

ANG1_BOVIN/26-143 DDYRYIHFLTQHYDAKPKGRNDEYCFNMMKNRRLTRPCKDRNTFIHGNKNDIKAICEDRNGC

Priority Order:
1 ~ 8 : DDYRYIHF
58 ~ 67 : DRNGQPYRGD
92 ~ 107 : RYGATEDSRVIVVGCE
21 ~ 34 : NDEYCFNMMKNRRL

Fig. 4. Chemical properties, including hydrophilicity, hydrophobicity, surface accessibility, polarity, secondary structure, etc., of each predicted linear epitope



http://140.121.196.30 - Remus 3D Display - Microsoft Internet Explorer

▼Protein 3D Structure ▼Sites(priority) ▼Control

Last updated: 06/20/2006 6:00pm

PDBFiles\1agi.pdb

Select PDB: PDBFiles\\1agi.pdb

□ 1onc.pdb
□ 1k2a.pdb_A
□ 1h1h.pdb_A
□ 1km9.pdb_A
□ 1h8x.pdb_A

Simutaneously Superimposed

☑ 🟥 1-8(1)
☑ 🟥 58-67(4)
☑ 🟥 92-107(16)
☑ 🟥 21-34(22)

Global Parameter:
□ Spin
Display: Ribbon Cartoon
Background Color: background white

Local Parameter:
Restrict selected Set Center
□ ShowLabel 14
Color: color aliceblue

Color meaning:
🟥 longer than the lower limit
🟩 shorter than the lower limit
🟨 trimmed residues
🟦 strictly fit in with a length longer than the lower limit
🟪 overlapping regions
🟪 residues of Loop

Start_End	Priority	Residues	Loop Ratio	Helix Ratio	Sheet Ratio
1_8	1	*DDYRYIHF*	50.000%	50.000%	0.000%
58_67	4	*DRNGQPYRGD*	70.000%	30.000%	0.000%
92_107	16	*RYGATEDSRVIVVGCE*	25.000%	0.000%	75.000%
21_34	22	*NDEYCFNMMKNRRL*	21.429%	78.571%	0.000%

[ILE]47.CA #405 網際網路

Fig. 5. 3D structural representation and detailed information of each identified linear epitope of the selected protein (PDB ID: 1agi:A, ANGI_BOVIN)

Acknowledgments. This work is supported by Center for Marine Bioscience and Biotechnology (CMBB), National Taiwan Ocean University and National Science Council (grants NSC 95-2221-E-019 -032 to Tun-Wen Pai, NSC95-2627-B007-003 and NSC95-2622-B-007-001-CC3 to Margaret Dah-Tsyr Chang) in Taiwan, R. O. C.

References

1. Chang, H.T., Pai, T.W., Fan, T.C., Su, B.H., Wu, P.C., Tan, C.Y., Chang, C.T., Liu, S.H., and Chang, M.D.T. (2006) A reinforced merging methodology for mapping unique peptide motifs in members of protein families. *BMC Bioinformatics* **7:38**.
2. Pai, T.W., Chang, M.D.T., Tzou, W.S., Su, S.H., Wu, P.C., Chang, H.T., and Chou, W.I. (2006) R$_E$MUS: a tool for identification of unique peptide segments as epitopes. *Nucleic Acids Res.* **34**, Web Server Issue. W198-201.
3. Mackay, I.R, Rowley, and M.J. (2004) Autoimmune epitopes: autoepitopes. *Autoimmun Rev.* **3**, 487-92.
4. Sharon, M., Rosen, O., and Anglister, J. (2005) NMR studies of V3 peptide complexes with antibodies suggest a mechanism for HIV-1 co-receptor selectivity. *Curr Opin Drug Discov Devel.* **8**, 601-12.
5. Schlessinger, A., Ofran, Y., Yachdav, G., amd Rost, B. (2006) Epitome: database of structure-inferred antigenic epitopes. *Nucleic Acids Res,* **34**, Database issue D777-780.
6. Kulkarni-Kale, U., Bhosle, S., and Klaskar, A.S. (2005) CEP:a conformational epitope prediction server. *Nucleic Acids Res,* **33**, Web server issue W168-171.

7. Roggen,E. (2006) Recent Developments with B-Cell Epitope Identification. *J. of Immunotoxicology*, 3:137-149.
8. Sollne, J., and Mayer, B. (2006) Machine learning approaches for prediction of linear B-cell epitopes on proteins. *J Mol Recognit.* **19**, 200-208.
9. Saha, S., and Raghava, G.P. (2006) AlgPred: prediction of allergenic proteins and mapping of IgE epitopes. *Nucleic Acids Res.* **34**, Web server issue, 202-209.
10. Alix, A.J. (1999) Predictive estimation of protein linear epitope by using the program PEOPLE,. *Vaccine*, **18**, 311-314.
11. Jones, S., and Thomton, J.M. (1997) Analysis of protein-protein interaction sites using surface patches. *J. Mol.Biol.*, **272**, 121-132.
12. Houslay, M.D., Schafer, P., and Zhang, K.Y. (2005) Keynote review: phosphodiesterase-4 as a therapeutic target. *Drug Discov. Today*, **10,** 1503-1519.
13. Finn, R., Mistry, J., Schuster-Böckler, B., Griffiths-Jones, S., Hollich, V., Lassmann, T., Moxon, S., Marshall, M., Khanna, A., Durbin, R., Eddy, S., Sonnhammer, E., and Bateman, A. (2006) Pfam: clans, web tools and services. *Nucleic Acids Res.* Database Issue **34**:D247-D251.
14. Kawashima, S. and Kanehisa, M. (2000), AAindex: Amino acid index database. *Nucleic Acids Res.* **28**:374.
15. Dayhoff, M.O. (1978) A model of evolutionary change in proteins. *Atlas of Protein Sequence and Structure*, **5**, Suppl 3, 345-352.
16. Henikoff, S., and Henikoff, J.G. (1992) Amino acid substitution matrices from protein blocks. *Proc. Natl. Acad. Sci. USA,* **89**, 10915-10919.

A Novel Greedy Algorithm for the Minimum Common String Partition Problem

Dan He

Department of Computer Science, University of Vermont
Burlington, VT 05405, USA

Abstract. The Minimum Common String Partition problem (MCSP) is to partition two given input strings into the same collection of substrings, where the number of substrings in the partition is minimized. This problem is a key problem in genome rearrangement, and is closely related to the problem of sorting by reversals with duplicates. MCSP is NP-hard, even for the most trivial case, 2-MCSP, where each letter occurs at most twice in each input string. There are various approximation algorithms which can achieve very good approximation ratios but with complicated implementations, for example, 1.5-approximation algorithm for 2-MCSP, 1.1037-approximation algorithm for 2-MCSP and a 4-approximation algorithm for 3-MCSP. There is also a simple greedy algorithm for MCSP which extracts the longest common substring from the given strings at each step. In this paper, we propose a novel greedy algorithm for MCSP, where we extract the longest common substring containing a symbol occurring only once at each step whenever there is a such symbol. We show our algorithm is more "worst case" greedy at each step than the greedy algorithm and the expected performance of our algorithm is better than that of the greedy algorithm. Our experiments show that our method achieves a better partition on average than the greedy algorithm does. Another advantage of our algorithm is that it is much faster than the greedy algorithm.

1 Introduction

A typical problem for string comparison is given a set of string operations (e.g., delete, insert, substitute, move a substring, reverse a substring), find the minimum number of operations needed to convert a string to the other. Edit distance and permutation sorting by reversals are two well known examples. The Minimum Common String Partition problem (MCSP) is to partition two given input strings into the same collection of substrings, where the number of substrings in the partition is minimized. This problem is motivated by genome rearrangement, and is closely related to the problem of sorting by reversals with duplicates.

A *partition* of a string S is a sequence $P = (P_1, P_2, ..., P_m)$ of substrings whose concatenation is S, namely $P_1P_2...P_m = S$. The substrings P_i are called *blocks* of P. Given two partitions P, Q for two strings S_1, S_2, respectively, the pair $\pi = (P, Q)$ is a *common partition* of S_1 and S_2 if P is a permutation of Q.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 441–452, 2007.

The *Minimum Common String Partition* problem (MCSP) is to find a common partition of two given strings S_1, S_2 with the minimum number of blocks. And we define *k-MCSP* the version of MCSP where each letter occurs at most k times in each input string. The optimal common partition is not necessarily unique.

The property *related*, namely each symbol has the same number of occurrences in two strings, is necessary and sufficient for these two strings to have common partitions. This property can be verified in linear time. And in the rest of the paper, we assume this property is held by the two input strings.

The motivation of the *MCSP* problem is mainly genome rearrangement applications. It was first introduced by Chen et al. [1]. They point out that *MCSP* problem is closely related to the problem of sorting by reversals with duplicates, a key problem in genome rearrangement. The size of the partition of two strings S_1 and S_2 can be considered as a novel distance measure between S_1 and S_2, as claimed by Chrobak et al.[3]. Besides the classical edit-distance measure which is defined as the number of three classical edit operations: insertion, deletion and substitution, there are various kinds of distance measures ([4], [5], [6], [7], [8], [9]) considering block operations in string comparison.

Lots of algorithms have been developed for MCSP. Cormode and Muthukrishnan [4] propose an $O(lognlog^*n)$-approximation algorithm for the edit distance with moves problem, a more general case of MCSP; Chen et al. [1] describe a 1.5-approximation algorithm for 2-MCSP; Goldstein et al. [2] give a 1.1037-approximation algorithm for 2-MCSP and a 4-approximation algorithm for 3-MCSP; Chrobak et al. [3] propose a 3-approximation greedy algorithm for 2-MCSP, which they called Algorithm Greedy. Algorithm Greedy also achieves an approximation ratio of at least $\Omega(logn)$ for 4-MCSP and an approximation ratio between $\Omega(n^{0.43})$ and $O(n^{0.69})$ for general MCSP. Chrobak et al. [3] claim that the implementation of the greedy algorithm, which extracts the longest common substring in each step, is considerably simple than the implementations of other approximation algorithms with tighter approximation ratios. Therefore the greedy algorithm is a likely choice for MCSP in many practical situations.

In this paper, we propose a novel greedy algorithm based on the longest common substring strategy. We first count the number of occurrences for each symbol in the strings and then extract the longest common substring containing a symbol with only one occurrence. If there is no such symbol, we apply the greedy strategy of Chrobak et al. [3] to extract the regular longest common substring. After each extraction we mark the substring extracted and reduce accordingly the number of occurrences of the symbols in the extracted substring. We apply this procedure iteratively until all the symbols are marked. We show the worst case scenario of our method is no worse than that of the greedy algorithm. And if there are such longest common substrings containing a symbol with only one occurrence, our method has a better worse case scenario than that of Algorithm Greedy. Our experiments on randomly generated strings show that our algorithm achieves a better partition on average than Algorithm Greedy does. Another advantage of our algorithm is that it is much faster than Algorithm Greedy.

The outline of this paper is as follows: In Section 2 we summarize previous greedy algorithm from Chrobak et al. [3] and present a novel greedy algorithm. We show the results of our experiments in Section 3. We include our final remarks in Section 4.

2 Iterative Greedy Algorithm

2.1 Iterative Greedy Algorithm

Chrobak et al. [3] propose a simple greedy algorithm for the MCSP which extracts a longest common substring from the given strings each step. They also claim that the idea of this greedy algorithm is simple and it is a good choice for solving MCSP in many practical situations. They prove that the approximation ratio of their greedy algorithm is 3 for 2-MCSP, $\Omega(logn)$ for 4-MCSP and between $\Omega(n^{0.43})$ and $O(n^{0.69})$ for general MCSP. They also give out an example where the Greedy algorithm fails to find the optimal partition. For two given strings $S_1 = cdabcdabceab$ and $S_2 = abceabcdabcd$, Algorithm Greedy returns the partition:

$$\langle (c, d, abcdabc, e, ab), (ab, c, e, abcdabc, d) \rangle$$

while the optimal partition is $\langle (cdabcd, abceab), (abceab, cdabcd) \rangle$.

Our Iterative Greedy algorithm is based on Algorithm Greedy of Chrobak et al. [3]. We build a profile for the number of unmarked occurrences of each symbol in the given strings. Then we extract the longest common substring containing a symbol with only one occurrence. If there is no such symbol, we apply the greedy strategy of Chrobak et al. [3] to extract the regular longest common substring. After each extraction we mark the substring extracted and reduce accordingly the number of occurrences of symbols which occur in the substring being extracted. We apply this procedure iteratively until all the symbols are marked. The algorithm is described in detail with pseudo-code in Figure 1.

Using the same example as in Chrobak et al. [3], we show that our Iterative Greedy algorithm can find the optimal common partition. Given two strings $S_1 = cdabcdabceab$ and $S_2 = abceabcdabcd$, Algorithm Greedy returns $\langle (c, d, abcdabc, e, ab), (ab, c, e, abcdabc, d) \rangle$. While according to our Iterative Greedy algorithm, we first build the profile of the number of occurrences for each unmarked unique symbol:

$$\langle (a - 3), (b - 4), (c - 3), (d - 2), (e - 1) \rangle$$

Therefore we pick symbol e and find the longest common substring containing e, namely $abceab$. After we mark the substring $abceab$ in both strings, we reduce the number of occurrences for each symbol accordingly:

$$\langle (a - 1), (b - 2), (c - 2), (d - 2), (e - 0) \rangle$$

Next we pick a and find the unmarked longest common substring containing a, namely $cdabcd$. Now there is no unmarked substrings. We then obtain

```
Iterative Greedy Algorithm
Assume $S_1$ and $S_2$ are two related input strings
Initialize all symbols in $S_1$ and $S_2$ to be unmarked
Build the profile of the number of unmarked occurrences and
their positions for each unique symbol in $S_1$ and $S_2$
While there are symbols in $S_1$ or $S_2$ unmarked {
  If there are symbols occurring only once {
    Pick up the first symbol $k$ occurring only once
    S $\leftarrow$ unmarked longest common substring of $S_1$ and $S_2$ containing symbol $k$
  }else{
    S $\leftarrow$ unmarked longest common substring of $S_1$ and $S_2$
   }
  Mark one occurrence of S in each of $S_1$ and $S_2$ as blocks
  Mark the corresponding occurrences in the profile for each symbol in $S$
  Update the corresponding numbers of unmarked occurrences for each symbol in $S$
}
 (P,Q) $\leftarrow$ sequence of consecutive marked blocks in $S_1$ and $S_2$
```

Fig. 1. Iterative Greedy Algorithm

the common partition $\langle(cdabcd, abceab), (abceab, cdabcd)\rangle$, which is the optimal common partition.

However, our method can not guarantee the optimality of the partition either. For example, given two strings *aabccbx* and *ccbaabx*, our method returns $\langle(aa, b, cc, bx), (cc, b, aa, bx)\rangle$, while the optimal partition is $\langle(aab, ccb, x), (ccb, aab, x)\rangle$. Although our method is not optimal either, we show in the next several subsections that the worst case scenario of our method is no worse than the Greedy algorithm and the expected performance of our algorithm is better than that of Algorithm Greedy.

2.2 Preliminaries

We use the same annotations by Chrobak et al. [3]. Given two related strings A and B, if π is a common partition of A and B, $\#blocks(\pi)$ is the number of blocks in π, which is also the *size* of π. A bijection ξ: $[n] \rightarrow [n]$ (where $[n] = \{1, 2, ..., n\}$) *preserves letters* of A and B if $b_{\xi_i} = a_i$ for all $i \in [n]$. A pair of consecutive positions $i, i+1 \in [n]$ is called a *break* of ξ if $\xi(i+1) \neq \xi(i)+1$. We use $\#breaks(\xi)$ to denote the number of breaks in ξ and we have $\#blocks(\xi) = \#breaks(\xi) + 1$. A substring $S = a_p a_{p+1}...a_{p+s}$ is written as set of indices $p, p+1, ..., p+s$, where $|S| = s+1$ is the *length* of S. If S is a common substring of A and B, S^A, S^B are then the occurrences of S in A, B respectively. For a common substring S of A and B, ξ *respects* S if it maps consecutive letters of $S^A = a_p a_{p+1}...a_{p+s}$ onto consecutive letters of $S^B = b_q b_{q+1}...b_{q+s}$, namely $\xi(i) = i + q - p$ for $i \in S^A$.

Chrobak et al. [3] introduce *reference common partition* of A and B that respects all the blocks $S_1, ..., S_{t-1}$ selected by Algorithm Greedy in steps 1 to t-1. Reference common partitions are used to estimate the "damage" caused by Algorithm Greedy. It may introduce more breaks and include more blocks gradually.

For $t = 0, 1, ..., g$ where g is the number of steps of Greedy on A and B, the *reference common partition* ρ_t is defined inductively as follows. Initially $\rho_0 = \pi$ where π is the minimum common partition and ρ_0 is the corresponding bijection of π. For any step $t = 1, .., g$, suppose $S_t^A = \{p, p+1, ..., p+s\}$ and $S_t^B = \{q, q+1, ..., q+s\}$. Function $\delta : S_t^A \to S_t^B$ is defined as $\delta(i) = i + q - p$ for $i \in S_t^A$. Then ρ_t is defined by [3]

$$\rho_t(i) = \min \begin{cases} \delta(i) & \text{for } i \in S_t^A \\ \rho_{t-1}(\delta^{-1}\rho_{t-1})^{l(i)}(i) & \quad \text{for } i \in [n] - S_t^A \end{cases} \tag{1}$$

where $l(i) = min\{\lambda \geq 0 : \rho_{t-1}(\delta^{-1}(\rho_{t-1}))^{\lambda}(i) \notin S_t^B\}$. So for $i \in [n] - S_t^A$, $\rho_t(i)$ is the first symbol not in S_t^B on the G-path (we give definition of G-path in the following paragraph) starting from i.

Chrobak et al. [3] introduces annotations of G-paths and G-cycles to analyze the introduction of new breaks. A bipartite graph $G \subseteq [n] \times [n]$ has edges $(i, \rho(i))$ for $i \in [n]$ (we call ρ-edges) and edges $(i, \delta(i))$ for $i \in S^A$ (we call δ-edges). Let $\overline{S}^A = [n] - S^A$ and $\overline{S}^B = [n] - S^B$. Without loss of generality, we assume S^A and $\overline{S}^A$ as the sets of nodes on the "left-hand" side of G and we assume S^B and $\overline{S}^B$ as the sets of nodes on the "right-hand" side. Then each node in $\overline{S}^A$ and $\overline{S}^B$ is incident to one ρ-edge and each node in S^A and S^B is incident to one ρ-edge and one δ-edge. Therefore G is a collection of vertex disjoint paths and cycles with edges alternate between ρ-edges and δ-edges. They are called G-paths and G-cycles. An example of G-paths is shown in the upper part of Figure 2.

Chrobak et al. [3] showed that each partition introduces at most 4 breaks.

Lemma 1. *[3] For each t=0,1,...,g, (a)ρ_t is a common partition of A, B, (b)ρ_t respects $S_1, ..., S_t$ and (c) if $t > 0$ then $\#breaks(\rho_t) \leq \#breaks(\rho_{t-1}) + 4$.*

Notice that in this lemma the information that S has maximum length is not used. So the construction of ρ_t can be used to convert any common partition π into another partition π' that has at most 4 breaks more than π. Chrobak et al. [3] show that new breaks can be introduced by the endpoints of S^A, the breaks inside S^A, and the endpoints of S^B.

Chrobak et al. [3] give an example shown in Figure 2. The upper part is reference common partition, which we call ρ, the lower part is new common partition after extracting the longest common substring $S^A = S^B = abccababd$. We call the new common partition ρ'. And we use ρ_A, ρ'_A to denote the partitions on the first string in ρ and in ρ', respectively, and use ρ_B, ρ'_B to denote the partitions on the second string in ρ and in ρ', respectively. As we can see, there are 4 more breaks in ρ', which are introduced by the two endpoints of S^A and the two endpoints of S^B: The break between block 7 and block 8 and the break between block 8 and block 9 in ρ'_A are introduced by the two endpoints of S^A. The break between block 1 and block 8 and the break between block 8 and block 6 in ρ'_B are introduced by the two endpoints of S^B. The two breaks inside S^A, which are the break between block 3 and block 4 and the break between block 4 and block 5 in ρ_A, also introduce two new breaks: the break between block

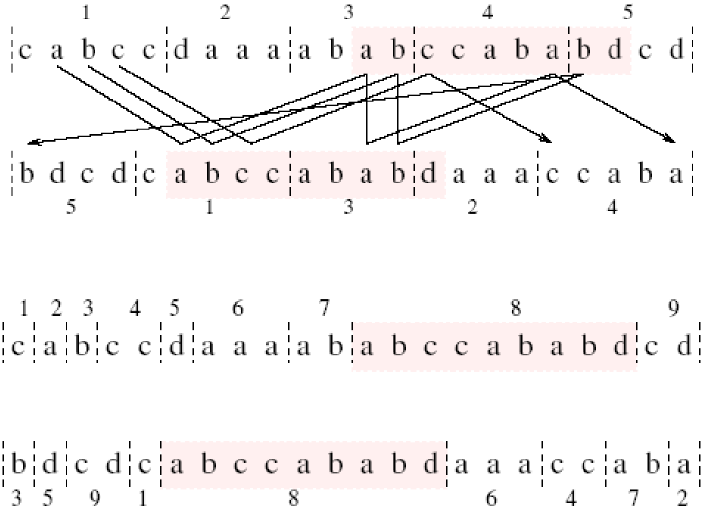

1 2 3 4 5
c a b c c d a a a a b a b c c a b a b d c d
b d c d c a b c c a b a b d a a a c c a b a
5 1 3 2 4
1 2 3 4 5 6 7 8 9
c a b c c d a a a a b a b c c a b a b d c d
b d c d c a b c c a b a b d a a a c c a b a
3 5 9 1 8 6 4 7 2

Fig. 2. The example given by Chrobak et al. [3]. The upper part is the reference common partition ρ and the lower part is the partition after extracting the unmarked longest common substring *abccababd*. The upper part also shows some G-paths.

3 and 4 and the break between block 2 and 3 in ρ'_A. However, these two new breaks are "compensated" by the covering of S^A on the two old breaks. So the total number of breaks increases only by 4.

However, the endpoints and the old breaks in S^A do not necessarily introduce new breaks in ρ'. If the endpoint of S^A is either the left symbol or the right symbol of a break in ρ_A (we simply say that the endpoint *is* a break), which indicates that S^A starts from or ends at the break in ρ, then the endpoint will not introduce new break. For example, in Figure 2, if we assume the lower part as a reference common partition and the unmarked longest common substring is *abccababd*, then the endpoints of this substring will not introduce new breaks. This is because the endpoints of this substring are also the breaks in the reference common partition and the substring does not cover any break and thus does not modify any block in the reference common partition. The same principle can be also applied to the endpoints of S^B.

For the old breaks in S^A, assume $i, i+1$ in S^A is a break in ρ, then $\xi(i+1) \neq \xi(i)+1$, where ξ is the bijection from A to B in ρ. Otherwise $i, i+1$ can not be a break since its corresponding bijection in B must also be a break. If $\delta(i) \neq \xi(i)$ and $\delta(i+1) \neq \xi(i+1)$, where δ is the bijection from S^A to S^B in ρ', this old break introduces a new break in ρ'. This means that if the bijections of both $i, i+1$ in ρ are different from the bijections of $i, i+1$ in ρ', the old break introduces a new break. This is shown in Figure 2, where the two old breaks in S^A introduce two new breaks. However, if one of the bijections of $i, i+1$ in ρ is the same as its corresponding bijection in ρ', namely if $\delta(i) = \xi(i)$ or $\delta(i+1) = \xi(i+1)$, then the old break does not introduce new break in ρ'.

Lemma 2. *Assume $i, i+1$ is a break in S^A of ρ. If $\delta(i) = \xi(i)$ or $\delta(i+1) = \xi(i+1)$, then the break $i, i+1$ in S^A of ρ does not introduce a new break of ρ'.*

Proof. We prove this lemma by contradiction. Assume the old break $i, i+1$ in S^A of ρ introduces a new break $j, j+1$ of ρ'_A. And without loss of generality, assume $\delta(i) = \xi(i)$. According to the definition of $\rho_t(i)$ in Equation (1) and the definition of G-path, and since $j, j+1 \in [n] - S^A$ is a new break in ρ'_A, there must be a G-path starting from j and passing through i. For example, in sequence A of the upper part of Figure 2, a G-path starts from a b in block 1 and passes through a b in block 3. Then there is a $k \in S^B$ such that $\delta^{-1}(k) = i$ and also there is a $f \in A$ such that $\xi(f) = k$, where k, f stand for positions. And we also have $\delta(i) = k$ in ρ' since δ is the bijection from S^A to S^B. Since $\delta(i) = \xi(i)$, we then have $\xi(f) = \xi(i)$. However, ξ is a bijection that preserves letters of A and B, there can not be $i \neq f$ such that $\xi(f) = \xi(i)$. Therefore the old break $i, i+1$ in S^A does not introduce a new break.

2.3 Worst Case Scenario

The worst case scenario for Algorithm Greedy is that the extraction of the longest common substring can introduce as bad as 4 more breaks. Chrobak et al. [3] show an example of the worst case scenario in Figure 2. As shown in Figure 2, the 4 more breaks come from the two endpoints of S^A and the two endpoints of S^B. The two old breaks in $S = abccababd$ also introduce to two new breaks in ρ' since the bijections of both sides of the two breaks in ρ are not equal to their corresponding bijections in ρ'. However, since S^A covers two old breaks, the total number of breaks is not increased by the introduction of the two new breaks.

While for our Iterative Greedy algorithm, if there is a symbol x occurring only once, the extraction of the longest common substring S containing x introduces at most 2 breaks in ρ'.

Lemma 3. *If there is a symbol x occurring only once, the extraction of the longest common substring S containing x introduces at most 2 breaks in ρ'.*

Proof. There are two cases:

1. x is an internal symbol of S
2. x is one of the endpoints of S

Since x occurs only once, there is only one way of bijection for it. Therefore $\xi(x) = \delta(x)$ (We call x is a *match symbol*, and if H is a substring and $\xi(a) = \delta(a)$ for all $a \in H$, we call H is a *match substring*). In case (1), since x is a match, there must be a common string f of length at least 1 containing x and f is a match substring(f can be x itself and then be a match symbol). Therefore the two breaks of the endpoints of f are the old breaks in S^A of ρ. Since the inner side of the two breaks are both match symbols, according to Lemma [2], these two breaks will not introduce new breaks. And since they are covered by S, the total number of breaks in ρ' are reduced by 2. So the number of breaks can be increased by at most $4 - 2 = 2$. In case (2), without loss of generality, let's assume x is the right endpoint of S. Since S is maximally extended, and S can not be right extended, and x is a match symbol, x then must also be a break in ρ and therefore it does not introduce a new break. Since S is a common string, the right endpoints of both S^A and S^B do not introduce new breaks. What's more, according to our analysis for case (1), there must also be a match substring f containing x. Therefore, the break of the left endpoint of f in S does not introduce any new break. Then the number of breaks can be increased by at most $4 - 2 - 1 = 1$.

One example for case (1) is the two given strings are $aabccbxbccbaa$ and $ccbaabxbaabcc$. Our Iterative Greedy algorithm returns $\langle(aa, b, cc, bxb, cc, b, aa), (cc, b, aa, bxb, aa, b, cc)\rangle$, while the optimal partition is $\langle(aab, ccb, x, bcc, baa), (ccb, aab, x, baa, bcc)\rangle$. x occurs only once and is a match symbol. The longest common substring containing x is $S = bxb$ and x is an internal symbol in S. The number of partition increases by only 2. Our example of two strings $aabccbx$ and $ccbaabx$ is in case (2). Our Iterative Greedy algorithm returns $\langle(aa, b, cc, bx), (cc, b, aa, bx)\rangle$, while the optimal partition is $\langle(aab, ccb, x), (ccb, aab, x)\rangle$. The symbol x occurs only once and is a match symbol. The longest common substring containing x is $S = bx$ and x is the right endpoint of S. The number of partition increases by only 1.

Since our Iterative Greedy algorithm extracts longest common substring containing a symbol occurring only once when there is such a symbol, and extracts regular longest common substring when there is no such a symbol, the worst case scenario of each step by our method, which introduces 2 more breaks, is no worse than the worst case sceneario by Algorithm Greedy, which introduces 4 more breaks. The expected performance of our Iterative Greedy algorithm is better than Algorithm Greedy since once there is a symbol occurring only once, our algorithm is guaranteed to introduce no more than 2 more breaks, while Algorithm Greedy may still introduce 4 more breaks. But since there are cases that in all steps no unmarked symbol occurring only once, our Iterative Greedy algorithm has the same approximation ratio as that of the Greedy Algorithm, which is $\Omega(3)$ for 2-MCSP, $\Omega(logn)$ for 4-MCSP and an approximation ratio between $\Omega(n^{0.43})$ and $O(n^{0.69})$ for general MCSP. Our experiments on hundreds of pairs of randomly generated strings indicate that the expected performance of our Iterative Greedy algorithm is better than that of Algorithm Greedy.

2.4 Time Complexity

Another advantage of our Iterative Greedy algorithm over Algorithm Greedy of Chrobak et al. [3] is the time complexity. In Algorithm Greedy, each step we need to extract the unmarked longest common substrings. This would take $O(n)$ time with a suffix tree, where n is the length of the given two strings. In our Iterative Greedy algorithm, we first build a profile for all the symbols in the given strings. The profile contains the numbers of unmarked occurrences for each symbol, and the positions of these occurrences. Building the profile takes $O(n)$ time. If there is no symbol occurring only once, we apply the greedy strategy to extract the regular longest common substring and the step takes $O(n)$ time too. However, if there are symbols occurring only once, our algorithm takes only $O(r)$ time to walk through the profiles and to find these symbols. Here r is the size of the alphabet of the unmarked substrings in the given strings. After we find the first symbol with only one unmarked occurrence, we just maximally extend the symbol to both directions to extract the longest common substring containing the symbol. The extraction of a substring would change the number of unmarked occurrences for those symbols in this substring and result in the reduction of r if the numbers of unmarked occurrences of some symbols become 0. And it is also possible that the numbers of unmarked occurrences of some symbols become 1, which can make our next step of extraction efficient. Although the time complexity depends on the quality of partitions, namely the number of steps, our experiments on randomly generated strings indicate that our Iterative Greedy algorithm is faster than Algorithm Greedy of Chrobak et al. [3].

3 Experiments

We conduct two sets of experiments on randomly generated strings and all the experiments are done on an Intel Celeron 1.8GHz processor with 256M memory.

In the first set of experiments, we generate random strings using similar way as that of Chen et al. [1]. The four components (n, r, k, t) denote the parameters of a pair of related strings. We first generate n distinct symbols as a string of length n, then we randomly pick two symbols and substitute all occurrences of them with a same new symbol. The process terminates until r singletons are left in the string. Therefore these are general $MCSP$ problems. We apply two operations on the strings. The first is k reversals whose endpoints are uniformly distributed with the size of the string. A reversal on substring $s[i], s[i+1], ..., s[j-1], s[j]$ is to reverse this substring to $s[j], s[j-1], ..., s[i+1], s[i]$. The second is t moves. A move is to randomly pick up a substring and move the substring to a random position. In our experiments, we set n as 2000, r as 500. Notice that this setting is reasonable since real genomes consists of lots of different genes and we consider each gene as a symbol here. So the number of different genes can be very large and 500 is reasonable. We first fix k as 40 and test the cases of t as 0, 20, 40, 60, 80, 100, respectively, and then fix t as 40 and test the cases of k as 0, 20, 40, 60, 80, 100, respectively. We apply both our Iterative Greedy algorithm and Algorithm Greedy of Chrobak et al. [3] on each (n, r, k, t) setting, and we test 10 independent instances of randomly generated strings for each parameter setting and show the average performances of each algorithm. In this set of experiments, the sizes of common partitions obtained by our Iterative Greedy algorithm and Algorithm Greedy are always the same, which indicates that for randomly generated strings, the Greedy algorithm performs well and it's hard

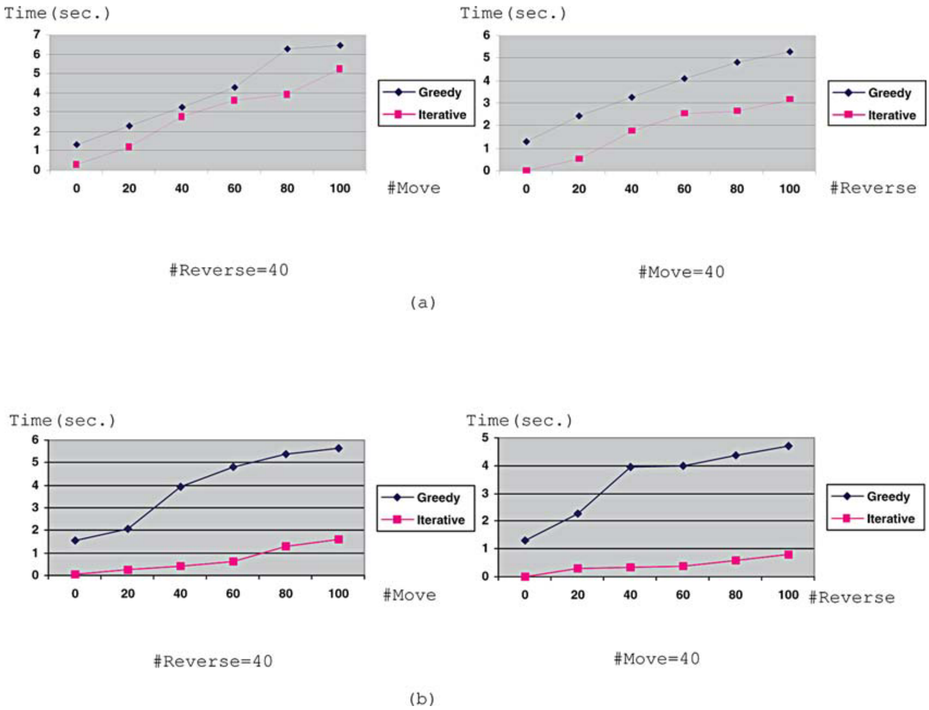


Fig. 3. The execution times our Iterative Greedy algorithm and the Greedy algorithm [3], for randomly generated strings without repeats (a) and with repeats (b), respectively

to generate cases where Iterative Greedy algorithm is better. However, as shown in Figure 3 (a), our Iterative Greedy algorithm is always faster than Algorithm Greedy.

In the second set of experiments, we randomly generate pairs of repeats (repeats like *abcdabcd*) and insert them into the string of length n. We set the occurrences of each symbol to be at most twice. Therefore this is 2-MCSP. We set the repeats to be no longer than 40. And we test the same parameter settings (n, r, k, t) as those in the first set of experiments. We also test different (k, t) pairs where $k, t \in \{0, 20, 40, 60, 80, 100\}$. We again apply our Iterative Greedy Algorithm and Algorithm Greedy of Chrobak et al. [3] on 10 instances of randomly generated strings and show the average performances of each algorithm. The experiment results are shown in Figure 3 (b) and Figure 4. The average performance of our Iterative Greedy algorithm is no worse than that of Algorithm Greedy for each parameter setting. As shown in Figure 4, our Iterative Greedy algorithm finds a better common partition in most of the cases. The most significant advantage of our algorithm over Algorithm Greedy we have obtained is 23, where $t = 100$ and $k = 0$. Also we can observe that the operation of

	#Move					
#Reverse						
	0	20	40	60	80	100
0	0	0	4	7	11	18
20	1	2	3	3	5	6
40	1	2	2	5	5	6
60	1	1	4	5	5	7
80	1	1	3	4	6	6
100	1	3	4	5	7	7

Fig. 4. The size of common partition by Algorithm Greedy over the corresponding size by our Iterative Greedy algorithm. The numbers of Reverses and Moves are set to be 0, 20, 40, 60, 80, 100, respectively. We show the average performance of 10 randomly instances for each Reverse-Move set.

Move affects the performance of our algorithm much more than Reverse. This is reasonable since Reverse would generate many blocks of length 1, therefore our Iterative Greedy algorithm would have no advantage over Algorithm Greedy. On the other hand, Move would generate long blocks where it is easy for our algorithm to make improvements. When there are many repeats, as shown in the example in Figure 2, our Iterative Greedy algorithm can perform much better. In this set of experiments, the execution time of our Iterative Greedy algorithm is much faster than that of Algorithm Greedy, as shown in Figure 3 (b). Compared with the first set of experiments, the execution time of Greedy algorithm doesn't change much, on the contrary, the execution time of our Iterative Greedy algorithm improves a lot. This is because the extraction of a substring would result in the number of unmarked occurrences of the symbols in this substring to be either 0 or 1, which makes our algorithm efficient. Therefore, as our experiments indicate, the repeats in the string not only improve the execution time of our Iterative Greedy algorithm, but also improve the quality of the common partition obtained by our algorithm. This set of experiments clearly indicates the expected performance of our Iterative Greedy algorithm is better than that of Algorithm Greedy.

4 Conclusion

In this paper, we propose a novel greedy algorithm for the minimum common string partition problem where we first try to extract the longest common substring containing a symbol occurring only once. We show that our Iterative Greedy algorithm has better expected performance than that of Algorithm Greedy by Chrobak et al. [3]. Our experiments show that this algorithm could achieve significant advantages over Algorithm Greedy in both time complexity and quality of the partition, especially when there are many repeats in the strings.

References

1. X. Chen, J. Zheng, Z. Fu, P. Nan, Y. Zhong, S. Lonardi and T. Jiang. Computing the assignment of orthologous genes via genome rearrangement. *Proc. of Asia Pacific Bioinformatics Conference 2005*, Jan 18-20, pp. 363-378, 2005.
2. A. Goldstein, P. Kolman, J. Zheng. Minimum common string partition problem: hardness and approximation. *Proc. of International Symposium on Algorithms and Computation (ISAAC04)*, pp.484-495, LNCS 3341, Hong Kong, China, 2004.
3. M. Chrobak, P. Kolman, J. Sgall. A greedy algorithm for the minimum common string partition problem. *Proc. 7th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'04)*, LNCS 3122, Springer 2004, pp. 84-95.
4. G. Cormode, J.A. Muthukrishnan. The string edit distance matching with moves. *Proc. 13th Annual Symposium on Discrete Algorithms (SODA)*, pp. 667-676, 2002
5. J.B. Kruskal and D. Snakoff. An anthology of algorithms and concepts for sequence comparision. *In Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, Edited by David Sankoff and Joseph B. Kruskal, Addison-Wesley, 1983.
6. D. Lopresti, A. Tomkins. Block edit models for approximate string matching. *Theoretical Computer Science 181*, pp. (159-179) 1997.
7. D. Shapira, J.A. Storer. Edit distance with move operations. *Proc. 13th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pp. 85-98, 2002.
8. W.F. Tichy. The string-to-string correction problem with block moves. *ACM Trans. Computer Systems 2*, pp. (309-321) 1984.
9. G.A. Watterson, W.J. Ewens, T.E.Hall, and A. Morgan. The chromosome inversion problem. *J. of Theoretical Biology*, 99: 1-7 1982.

An Efficient Algorithm for Finding Gene-Specific Probes for DNA Microarrays*

Mun-Ho Choi, In-Seon Jeong, Seung-Ho Kang, and Hyeong-Seok Lim

Dept. of Computer Science, Chonnam National University,
Yongbong-dong 300, Buk-gu, Gwangju 500-757, Korea
howork@paran.com, isjung0@hotmail.com,
kinston@natural.chonnam.ac.kr, hslim@chonnam.ac.kr

Abstract. The accuracy of a DNA microarray is fairly dependent on the quality of the probes it uses; a good probe should be specific for exactly one gene. Most sequence based algorithms use the edit distance to the target sequences as the measure of the specificity of the probe. We propose a novel algorithm for finding gene-specific probes which avoids large amounts of redundant computations of the edit distance, while maintaining the same accuracy as that provided by an exhaustive search. Our approach utilizes the fact that when the starting position of a probe candidate is moved only a few base pairs, the change in the edit distance to the off-target sequence is limited. The proposed algorithm does not use any index structures and is insensitive to the length of the probes. Our approach enables short (20∼30 bases) or long (50 or more bases) probes to be computed for genomes of size 10M within a day.

1 Introduction

The DNA microarray is a widely used tool to perform rapid experiments on a large scale in areas such as gene discovery and mapping, gene regulation studies, diagnosis, drug discovery, and toxicology[13]. In general, microarrays have been constructed with two types of probes, PCR-generated probes that typically range in size from 200 to 2,000 base pairs and oligonucleotide probes that are between 20 and 70 nucleotides long[4].

The accuracy of a microarray is fairly dependent on the quality of the selected probes. Good probes (*homogeneity*) should have similar reaction conditions, (*sensitivity*) should not form stable secondary structures that may interfere with the probes in forming heteroduplexes during hybridization, and (*specificity*) should be totally specific to their respective targets to avoid any cross-hybridization.

Empirically, the optimum probe for a gene would be the one with the minimum hybridization free energy for the target gene and the maximum hybridization free energy for all other genes in the hybridizing pool. Thermodynamic approaches are used to calculate the hybridization energy and to predict secondary structures[22,3].

* This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)(KRF-2005-041-D00747).

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 453–464, 2007.

Given the current uncertainties in the hybridization energy in a chip environment[3], most biologists use the working hypothesis that the specificity of a probe can be well approximated by the dissimilarity to the target sequences[12]. For the sake of accuracy and convenience, most researchers use the edit distance as a dissimilarity measure.

Searching for similar patterns in DNA sequences can be considered as the traditional approximate string matching problem. There exist numerous algorithms that conduct approximate searches using the edit distance[16], the fastest of which is based on so-called bit-parallelism[15,8].

Another approach to accelerate approximate string matching is filtering[6]. To filter out uninteresting parts of text in a reasonable time, index structures for the text, such as a suffix tree or q-grams, are used[16]. However, in the context of probe design, the existing indexing schemes do not perform well in practice, because genome data are very large and the size of the DNA alphabet is only four. Some index structures require a large amount of memory (several times the text size), while others suffer from a low filtering ratio[16].

Currently, for the inspection of the probe specificity, in computational biology, two approaches are commonly used. Some applications perform a homology search with BLAST. This approach uses the heuristics to speed up the search; but cannot guarantee that all significant matches will be found[18]. Other applications use similarity search algorithms based on the filtering scheme that does not lose any significant matches[6].

In this paper, we propose a novel filtering approach which does not use any index scheme. Our approach takes advantage of the intra-similarities between the probe candidates. Typically, all consecutive substrings of the gene sequence are potential probe candidates, thus, the computation time can be reduced by utilizing the fact that when the starting position of a probe candidate is moved only a few base pairs, the change in the edit distance to the off-target sequence is limited.

2 Preliminary and Related Work

We will use the following notation with strings. We assume that strings are sequences of characters from a finite character set Σ. The ith character of a string s is denoted by s_i, and $s_{i..j}$ denotes the substring of s that begins at the ith position and ends at the jth position. The first character has index 1, and so $s = s_{1..|s|}$. In particular, if $j < i$, then $s_{i..j} = \epsilon$, an empty string; if $i \leq 0$, then $s_{i..j} = s_{1..j}$; if $j > |s|$, then $s_{i..j} = s_{i..|s|}$. We will use the notation st to denote the concatenation of the strings s and t.

Approximate string matching. The approximate string matching problem can be stated as follows. Given a text T of length n, a pattern P of length m, and an integer k, find all text positions j such that the suffix of T matches P, allowing at most k errors(differences). We use the term *error level* to refer to $\alpha = k/m$.

The *distance* determines how different two strings are. The edit distance between two strings, x and y, is defined as the number of edit operations that transform x into y or vice versa. The most common form of edit distance is the *Levenshtein edit distance*, for which the allowed edit operations are insertions, deletions, and substitutions of a single character. Another choice of distance is the *Hamming distance* which allows only replacements. We will use $ed()$ to denote the Levenshtein edit distance.

The canonical dynamic programming (in short DP) algorithm for the determination of the edit distance between a text $T_{1..n}$ and a pattern $P_{1..m}$ computes an $(m+1)\times(n+1)$ DP matrix $D[0..m, 0..n]$ in time $O(mn)$ using the recurrence relation

$$D[i,j] = \min \left\{ \begin{array}{l} D[i-1,j-1] + \delta_{ij} \\ D[i-1,j] + 1 \\ D[i,j-1] + 1 \end{array} \right\}, \text{ where } \delta_{ij} = \begin{cases} 0, & \text{if } P_i = T_j \\ 1, & \text{otherwise.} \end{cases}$$

For the global edit distance problem, the base conditions are $D[i,0] = i$ and $D[0,j] = j$. $ed(T_{1..j}, P_{1..i}) = D[i,j]$, and so $ed(T_{1..n}, P_{1..m}) = D[m,n]$.

We denote the edit distance of approximate string matching as $ed_A()$, for which the base conditions are $D[i,0] = i$ and $D[0,j] = 0$, since an occurrence can start anywhere in the text. The edit distance of approximate string matching between a text $T_{1..n}$ and a pattern $P_{1..m}$ is $ed_A(T_{1..n}, P_{1..m}) = \min_{0\le j\le n}\{D[m,j]\}$, since an occurrence can end anywhere in the text.

The DP algorithm was refined to $O(kn)$ by Ukkonen[24] in the case where only the hits of a pattern in a text with at most k mismatches are significant. The speed of the canonical algorithm was further increased using bit-parallelism[15,8]. The key idea is to use the differences between the entries of the DP matrix instead of their absolute values (see Property 1) in order to take advantage of the intrinsic parallelism of the bit operations inside a computer word and so reduce the number of operations. The number of operations that an algorithm performs can be reduced by a factor of w, where w is the number of bits in a computer word. This technique permits a running time of $O(\lceil m/w \rceil\, n)$ to be achieved.

Property 1. *[23] In the DP matrix, for all $1 \le i \le m$ and $1 \le j \le n$,*

(a) $\Delta h_{i,j} = D[i,j] - D[i,j-1] \in \{-1, 0, +1\}$,
(b) $\Delta v_{i,j} = D[i,j] - D[i-1,j] \in \{-1, 0, +1\}$,
(c) $\Delta d_{i,j} = D[i,j] - D[i-1,j-1] \in \{0, +1\}$.

Another approach to accelerate approximate string matching is filtering. General filtering criteria are based on the Pigeonhole principle. If T matches pattern P with k errors, and $P = z_1 z_2 ... z_j$(a concatenation of subpatterns), then some substring of T matches at least one of the z_i's, with $\lfloor k/j \rfloor$ errors[1]. Based on this condition, if we divide a pattern into $k+1$ non-overlapping subpatterns, then at least one of the subpatterns has to match without error. Thus, text areas that have no perfect match to then subpatterns can be discarded from further consideration.

The performance of filtering algorithms, however, is very sensitive to the error level, since it affects the amount of text that has a potential match. Also, a filtering algorithm is normally unable to discover the matching text positions by itself and, consequently, these potential match positions then need to be verified with another algorithm(e.g., bit-parallel algorithm).

For filtering algorithms to run in a reasonable amount of time for a long text, they use an index scheme. Indexing based on a suffix tree requires an unrealistically large amount of memory (12 to 70 times the text size). Other algorithms based on q-grams have reasonable memory requirements, but work well with only a small number of allowed mismatches. The suffix array, which strikes a compromise between the required memory and search time, has been shown to be useful, but still requires a large amount of memory(four times the text size)[17]. One important class of algorithms used for this problem consists of those that are hybrid in the sense that they reap the benefits of suffix tree traversal and filtering based on q-grams[14,17,7,11].

Probe design problem. In general, multiple probes per gene are used in designing microarrays, in order to obtain reliable quantitative information of gene expression[5,2]. Thus, lots of probe candidates have to be generated. Typically, all consecutive substrings of the gene sequence are potential probe candidates and the probe design problem can be solved based on probe candidates elimination. Initially, for every gene g, every segment(substring) of gene g whose length is l is assumed to be feasible as a probe. “Bad” probe candidates are filtered out using the three criteria: homogeneity, sensitivity, and specificity. Specificity filtering is the most time consuming phase because the dissimilarities between all probe candidates and all other genes in the genome should be checked.

There exist many probe design tools (see ref. [18]). To determine the specificity of probe candidates, several tools (OligoArray, OligoWiz, ROSO, OligoPicker, Yoda) rely on BLAST or an alternative which is similar to BLAST. The others perform specificity filtering by themselves based on approximate pattern matching. Some of them (ProbeSelect, PROBESEL) use a filtering algorithm with a suffix array. Hyyrö *et al.*[10] used an 8-gram indexing scheme with 1-neighborhood generation. Rahmann[19,20] used a suffix array with the longest common substring instead of the edit distance. Sung and Lee[21] used a gapped hashing method with the Hamming distance as a similarity measure.

3 Problem Definition and Our Approach

Given a certain gene, we aim to identify all probes that can be obtained from it such that no other genes in the genome are approximately matched to them. The more strict description of specificity filtering is: given a set of genes, G, for each probe candidate p of length l in a gene g, find out whether there exists a substring q in $G - \{g\}$ such that $ed(p, q) \leq k$, for some threshold k. If such a q is found, then the probe p is said to be able to cross-hybridize with other genes and, thus, it is not a “good” probe. The aim of specificity filtering is to filter out all such “bad” probes.

```
Algorithm 1. ClassicalApproximateSearch
input: text $T$, pattern $P$, substring length $l$, maximal number of errors $k$
output: Set $S$ of pairs of $(i, j)$ such that $ed(T_{t..j}, P^i) \leq k, 1 \leq t \leq j$
    $S \leftarrow \emptyset$
    for $i \leftarrow l$ to $m$ do
        $S \leftarrow S \cup \{\{i\} \times ApproximateStringMatching(T, P^i, k)\}$
```

Fig. 1. The classical approximate string matching algorithm

Problem Definition. Given a text $T_{1..n}$, a pattern $P_{1..m}$, a pre-specified substring length l, and a maximal number of errors k, $0 \leq k < l$, find all text positions j for all subpatterns $\{P^i | P^i = P_{i-l+1..i}, l \leq i \leq m\}$ of length l, such that there exists t such that $ed(T_{t..j}, P^i) \leq k$, $1 \leq t \leq j$.

Hereafter, to denote the whole text and pattern, we use T and P, respectively, and P^i and T^j to denote the substring of P of length l such that $P^i = P_{i-l+1..i}$ and the substring of T of length $2l$ such that $T^j = T_{j-2l+1..j}$, respectively.

Fig. 1 represents a classical solution for the gene specific probe design. For the set of genes, G, the text T is the concatenated string of all gene sequences, such that $T = g_1\$g_2\$...\$g_{|G|}, \$ \notin \Sigma$. The pattern P is the DNA sequence of gene $g \in G$ for which we aim to design probes.

If the returned set S contains any pair (i, j) and j is not a position which corresponds to the gene g, then the probe candidate P^i is "bad", and so it is filtered out. The function *ApproximateStringMatching* returns a set of all positions in the text where the given pattern occurs allowing up to k mismatches. The time complexity of Algorithm 1 is $O(\lceil l/w \rceil mn)$, since *ApproximateStringMatching* requires a running time of $O(\lceil l/w \rceil n)$, where w is the number of bits in a computer word.

Let us introduce another method that obtains the same results as Algorithm 1 by comparing each subpattern P^i to all substrings of text T whose length is $l + k$.

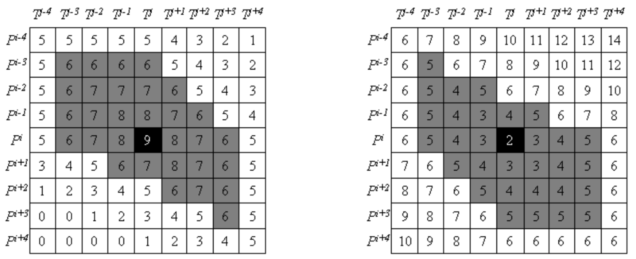
Lemma 2. *Given a text $T_{1..n}$, and a pattern $P_{1..l}$, all text positions j such that there exists t such that $ed(T_{t..j}, P_{1..l}) \leq k, 1 \leq t \leq j$, are*

$$\{j | ed_A(T_{j-l-k+1..j}, P_{1..l}) \leq k, 1 \leq j \leq n\}$$

for all $0 \leq k \leq l$.

Proof. If an occurrence of P with up to k mismatches ends at j, then the occurrence starts, at least, at $j - l - k + 1$ since there are at most k insertions of P to make it match with T. So all occurrences of P up to k mismatches can be found by using $ed_A(T_{j-l-k+1..j}, P_{1..l})$. □

The following lemma provides logical bases of filtering: if any occurrence of a pattern P^i appears in a text T at a position j with errors $k' \gg k$, then, for the small natural numbers t_1 and t_2, the pattern $P^{i \pm t_1}$ is rarely matched with the text $T^{j \pm t_2}$ while allowing for up to k errors.



	T^{j-4}	T^{j-3}	T^{j-2}	T^{j-1}	T^{j}	T^{j+1}	T^{j+2}	T^{j+3}	T^{j+4}
P^{i-4}	5	5	5	5	5	4	3	2	1
P^{i-3}	5	6	6	6	6	5	4	3	2
P^{i-2}	5	6	7	7	7	6	5	4	3
P^{i-1}	5	6	7	8	8	7	6	5	4
P^{i}	5	6	7	8	9	8	7	6	5
P^{i+1}	3	4	5	6	7	8	7	6	5
P^{i+2}	1	2	3	4	5	6	7	6	5
P^{i+3}	0	0	1	2	3	4	5	6	5
P^{i+4}	0	0	0	0	1	2	3	4	5

(a) the lower bounds deduced from $ed_A(T^j,P^i)$=9

	T^{j-4}	T^{j-3}	T^{j-2}	T^{j-1}	T^{j}	T^{j+1}	T^{j+2}	T^{j+3}	T^{j+4}
P^{i-4}	6	7	8	9	10	11	12	13	14
P^{i-3}	6	5	6	7	8	9	10	11	12
P^{i-2}	6	5	4	5	6	7	8	9	10
P^{i-1}	6	5	4	3	4	5	6	7	8
P^{i}	6	5	4	3	2	3	4	5	6
P^{i+1}	7	6	5	4	3	3	4	5	6
P^{i+2}	8	7	6	5	4	4	4	5	6
P^{i+3}	9	8	7	6	5	5	5	5	6
P^{i+4}	10	9	8	7	6	6	6	6	6

(b) the upper bounds deduced from $ed_A(T^j,P^i)$ =2

Fig. 2. Example of the lower and upper bounds of edit distances deduced from $ed_A(T^j, P^i)$

Lemma 3. *For a text T and a pattern P*

(a) $ed_A(T^{j\pm1}, P^i) - ed_A(T^j, P^i) \in \{-1, 0, +1\}$,
(b) $ed_A(T^{j\pm1}, P^{i\pm1}) - ed_A(T^j, P^i) \in \{-1, 0, +1\}$,
(c) $ed_A(T^j, P^{i+1}) - ed_A(T^j, P^i) \in \{-2, -1, 0, +1\}$ *and*
$ed_A(T^j, P^{i-1}) - ed_A(T^j, P^i) \in \{-1, 0, +1, +2\}$.

Proof. Let $P^i = ap$, $P^{i+1} = pb$, $T^j = cq$, and $T^{j+1} = qd$; p is a common substring of P^i and P^{i+1}, and q is a common substring of T^j and T^{j+1}.

(a) Due to the property of DP matrix(Property 1(a)) $ed_A(cqd, P^i) - ed_A(cq, P^i) \in \{-1, 0, +1\}$. And, from Lemma 2, $ed_A(cqd, P^i) = ed_A(cq, P^i)$ or $ed_A(cqd, P^i) = ed_A(qd, P^i)$ since $ed_A(cqd, P^i) = \min\left\{ed_A(cq, P^i), ed_A(qd, P^i)\right\}$. So Lemma 3(a) holds.

(b) We first prove the subcase $ed_A(T^{j+1}, P^{i+1}) - ed_A(T^j, P^i) \in \{-1, 0, +1\}$. We obtain $ed_A(qd, pb) - ed_A(q, p) \in \{0, +1\}$ and $ed_A(cq, ap) - ed_A(q, p) \in \{0, +1\}$ from Property 1(c). Thus, $ed_A(qd, pb) - ed_A(cq, ap) \in \{-1, 0, 1\}$. The remaining case can be proved in the same way.

(c) We can show that $ed_A(T^j, P^{i+1}) - ed_A(T^j, P^i) \in \{-2, -1, 0, +1, +2\}$ in the same way as we did in (b) by using Property 1(b). Now, we prove that $ed_A(T^j, P^{i+1}) - ed_A(T^j, P^i) \neq 2$. Let $e = ed_A(T^j, ap)$. Suppose that $ed_A(T^j, pb) = e+2$. Due to Property 1(b), $ed_A(T^j, apb) \leq e+1$. We can divide T^j into U and V such that $T^j = UV$ and $ed_A(U, a) + ed_A(V, pb) = e+1$. $ed_A(V, pb) \leq e+1$ since $ed_A(U, a) \geq 0$. Because V is a suffix of T^j, $ed_A(T^j, pb) \leq e+1$, however, this contradicts with $ed_A(T^j, pb) = e+2$. Thus, $ed_A(T^j, pb) \neq e+2$. The remaining case can be proved in the same way. □

For the given $ed_A(T^j, P^i)$, the lower bounds and upper bounds of the edit distances between the patterns near to P^i and the texts near to T^j are determined according to Lemma 3. In Fig. 2, for example, the values of the cells are lower bounds or upper bounds deduced from the dark cells. If the maximal number of errors, k, is 5 then the values of the gray cells guarantee that the corresponding probe candidates and the targets (a) have no potential match or (b) are matched while allowing for up to k errors.

Corollary 4. *For a given maximal number of errors, k, an integer h, $0 \leq h$, d_L and d_H such that $d_L = ed_A(T^j, P^i) - k$ and $d_H = ed_A(T^{j+h}, P^{i+h}) - k$*

(a) if $d_L > 0$, $d_H > 0$, and $d_L + d_H \geq h + 1$,
then $ed_A(T^{j+t}, P^{i+t}) > k$ for all t such that $0 \leq t \leq h$, and
(b) if $d_L \leq 0$ and $d_H \leq 0$, and $|d_L| + |d_H| \geq h - 1$,
then $ed_A(T^{j+t}, P^{i+t}) \leq k$ for all t such that $0 \leq t \leq h$.

Proof. This corollary can be directly proved from Lemma 3. □

From Corollary 4, the interval $[i, i+h]$ is said to be either (case (a)) *over-covered* or (case (b)) *under-covered* by d_L and d_H, or otherwise, *uncovered*. The covered intervals can be discarded from further inspection.

Fig. 3 presents a jumping approximate search algorithm which applies Corollary 4. The function *ApproximateStringMatchingEx* does the same thing as the *ApproximateStringMatching* of Algorithm 1, but it returns the last row of the DP matrix after extending both end sides as follows: $D[m, l - t] = \max(D[m, l] - t, k + t)$ and $D[m, n + t] = \max(D[m, n] - t, k + t)$ for $0 < t \leq h$.

The function *InspectRangeCovering* examines the edit distances in U and V, that are returned by *ApproximateStringMatchingEx*, to determine whether there is an uncovered interval. If there is an uncovered rectangle, i.e. the rectangle enveloping consecutive uncovered intervals(e.g. the area $[j'_L, j'_H] \times [i_L, i_M]$ of Algorithm 2), the rectangle is divided into two half-sized rectangles, the new edit distance vector is computed, and then *InspectRangeCovering* is called recursively for each half-sized rectangle.

Fig. 4 shows an example of the jumping approximate search method. The first and last rows denote the extended vectors, U and V, that are returned by *ApproximateStringMatchingEx* taking P^{16} and P^{20} as patterns, respectively. The extended values are circled. The white bars indicate that, from those positions, there is a series of uncovered intervals. The gray bars indicate that, from those positions, there is a series of covered intervals. Only the boxed areas are uncovered, so, further inspections of them need to be done.

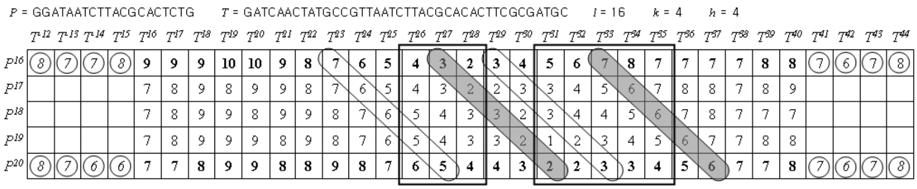
In contrast to Algorithm 1, only every hth probe candidate is compared to off-targets by computing the edit distance to the off-targets. h is referred to as the *jump*. The amount of discarded areas is influenced by the jump. A bigger jump leads to a higher filtering ratio, however too big a jump can lead to a large amount of uncovered areas, thereby reducing the filtering effects.

In the context of probe design, the optimal jump h is larger than $\lfloor l/2 \rfloor - k$ for the moderate error level α(see experimental results). Also, in the case of the optimal jump, very few rectangles remain uncovered when *InspectRangeCovering* is called directly from the main procedure. Thus, the time complexity of Algorithm 2 is asymptotic to $O(\lceil l/w \rceil \lceil m/h \rceil n)$, where w is the number of bits in a computer word.

Algorithm 2 is not only simple and efficient, but also provides many preferable features. This algorithm is itself a verification algorithm, it needs no preprocessing and does not use any index scheme. Because the algorithm can be applied gene by gene, only a very small amount of additional memory is required. In

```
**Algorithm 2.** $JumpingApproximateSearch$
input: text $T$, pattern $P$, substring length $l$, maximal number of errors $k$, jump $h$
output: Set $S$ of pairs of $(i, j)$ such that $ed_A(T^j, P^i) \leq k$
begin
01    $S \leftarrow \emptyset$;    $i_L \leftarrow l$
02    $U \leftarrow ApproximateStringMatchingEx(T, P^{i_L}, h)$
03    while $i_L < |P|$ do
04        $i_H \leftarrow \min(i_L + h, |P|)$
05        $V \leftarrow ApproximateStringMatchingEx(T, P^{i_H}, h)$
06        $InspectRangeCovering(l, |T|, i_L, i_H, U, V)$
07        $i_L \leftarrow i_H$;    $U \leftarrow V$
end.
function $InspectRangeCovering(j_L, j_H, i_L, i_H, U, V)$
begin
08    $h' \leftarrow i_H - i_L$
09    if $h' = 1$ then
10        $S \leftarrow S \cup \{(i_L, t) | U[t - j_L] \leq k, j_L \leq t \leq j_H\}$
11        $S \leftarrow S \cup \{(i_H, t) | V[t - j_L] \leq k, j_L \leq t \leq j_H\}$
12    $state \leftarrow CLOSED$;    $j \leftarrow j_L - h'$
13    while $j \leq j_H$ do
14        $d_L \leftarrow U[j - j_L] - k$;    $d_H \leftarrow V[j + h' - j_L] - k$
15        if interval$[i_L, i_H]$ is *over-coverd* by $d_L$ and $d_H$ then
16            if $state = OPENED$ then
17                $j'_H \leftarrow j + h' - d_H$;    $state \leftarrow CLOSED$
18                $i_M \leftarrow (i_L + i_H)/2$
19                $Z \leftarrow ApproximateStringMatchingEx(T_{j'_L..j'_H}, P^{i_M}, h')$
20                $InspectRangeCovering(j'_L, j'_H, i_L, i_M, U, Z)$
21                $InspectRangeCovering(j'_L, j'_H, i_M, i_H, Z, V)$
22        else if interval$[i_L, i_H]$ is *under-coverd* by $d_L$ and $d_H$ then
23            $S \leftarrow S \cup \{(i_L + t, j + t) | 0 \leq t \leq h'\}$
24            if $state = OPENED$ then
25                $j'_H \leftarrow j + h' - |d_H| - 1$;    $state \leftarrow CLOSED$
26                $S \leftarrow S \cup \{(i_H - t', j + h' - t) | 1 \leq t \leq |d_H|, 0 \leq t' < t\}$
27                Do same as the lines from 18 to 21.
28        else
29            if $state = CLOSED$ then
30                $state \leftarrow OPENED$
31                if $d_L > 0$ then $j'_L \leftarrow j + d_L$
32                else $j'_L \leftarrow j + |d_L| + 1$
33                    $S \leftarrow S \cup \{(i_L + t', j + t) | 0 \leq t \leq |d_L|, 0 \leq t' \leq t\}$
34        $j \leftarrow j + 1$
35    if $state = OPENED$ then
36        $j'_H \leftarrow j_H$
37        Do same as the lines from 18 to 21.
38    return $S'$
end.
```

Fig. 3. The jumping approximate string matching algorithm



P = GGATAATCTTACGCACTCTG T = GATCAACTATGCCGTTAATCTTACGCACACTTCGCGATGC l = 16 k = 4 h = 4

	T^{12}	T^{13}	T^{14}	T^{15}	T^{16}	T^{17}	T^{18}	T^{19}	T^{20}	T^{21}	T^{22}	T^{23}	T^{24}	T^{25}	T^{26}	T^{27}	T^{28}	T^{29}	T^{30}	T^{31}	T^{32}	T^{33}	T^{34}	T^{35}	T^{36}	T^{37}	T^{38}	T^{39}	T^{40}	T^{41}	T^{42}	T^{43}	T^{44}
P^{16}	8	7	7	8	9	9	9	10	10	9	8	7	6	5	4	3	2	3	4	5	6	7	8	7	7	7	7	8	8	7	6	7	8
P^{17}					7	8	9	8	9	9	8	7	6	5	4	3	2	2	3	3	4	5	6	7	8	8	7	8	9				
P^{18}					7	8	9	9	8	9	9	8	7	6	5	4	3	3	2	3	4	4	5	6	7	8	7	7	7				
P^{19}					7	8	9	9	9	8	9	8	7	6	5	4	3	3	2	1	2	3	4	5	6	7	7	8	8				
P^{20}	8	7	6	6	7	7	8	9	9	8	8	9	8	7	6	5	4	4	3	2	2	3	3	4	5	6	7	7	8	7	6	7	8

Fig. 4. Example of jumping approximate search

contrast to other filtering methodologies, this algorithm not only discards those areas that have no potential match, but also excludes those areas that preserve approximate matches from further inspection: the under-covered rectangles that preserve approximate matches are skipped with no verification.

In addition, the algorithm is not sensitive to the length of the probes. Although a longer probe causes the computing time of the edit distance to be longer, a larger edit distance leads to a bigger jump. This makes the algorithm insensitive to the length of the probe. In fact, a shorter running time is required for a longer probe length(see experimental results).

The proposed approach can be adapted to perform an approximate search which adopts the Hamming distance instead of the Levenshtein edit distance. In the case of the Hamming distance, only Lemma 3(b) holds.

4 Experimental Results

The proposed algorithms are implemented in C language[1] and tested on a single 32-bit processor system (Pentium-4 3.2Ghz with 512M RAM) and 64-bit processor system (dual Pentium-4 3.2Ghz with 4G RAM). The function *ApproximateStringMatchingEx* of Algorithm 2 is implemented according to [8]. The genomes involved in the experiments are listed in Table 1.

Table 1. Information of datasets used in experiments

	E. coli	S. pombe	S. cerevisiae	N. crassa
Length (bps)	4,846,583	7,278,949	8,865,725	16,534,812
# of genes	5,589	5,487	5,869	10,074

A summary of the running times is shown in Table 2. Li and Stormo used the fact that, in general, the hybridization free energy of a near-match is sufficiently large when the near-match contains more than 4 errors for 25mer oligonucleotides and 10 errors for 50mer oligonucleotides[12]. Hyrrö *et al* tested numerous approximate string matching methods and presented the results of an exhaustive

[1] We used the gcc compiler with full optimization (`-fexpensive-optimizations`) and the `long long` type is used for the 64bit words. The gcc compiler provides internal emulations of the `long long` type for the 32bit architecture.

Table 2. Running times of Algorithm 2

		E. coli		S. pombe		S. cerevisiae		N. crassa	
		l=25	l=50	l=25	l=50	l=25	l=50	l=25	l=50
		k=4	k=10	k=4	k=10	k=4	k=10	k=4	k=10
running time (min)	tested on 32bit system	314	356	698	790	1,221	1,145	4,644	4,927
	tested on 64bit system	445	201	894	408	1,601	802	5,703	2,795

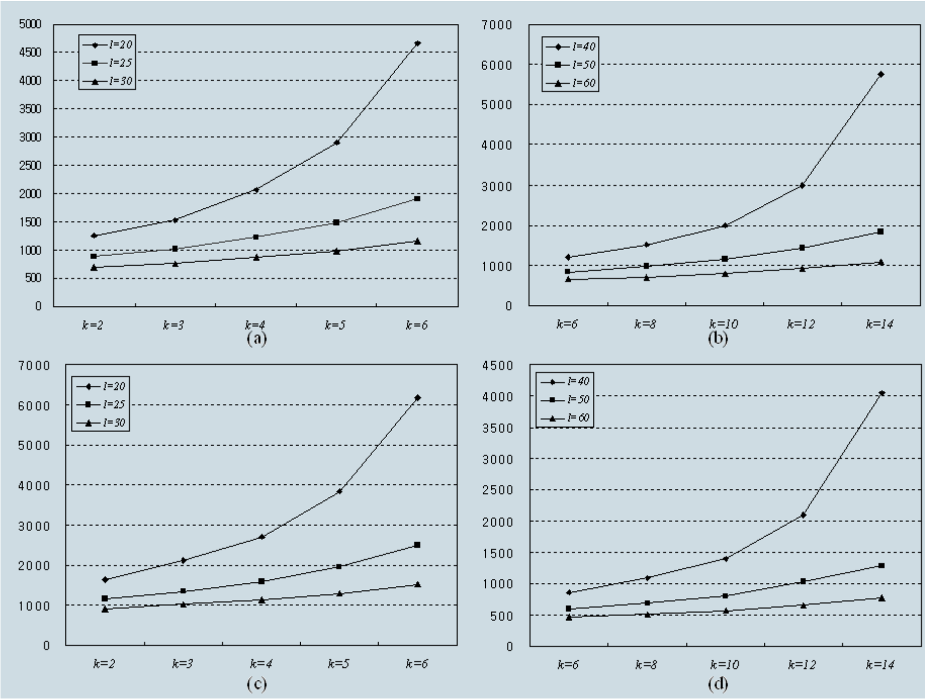


Fig. 5. Running time of the exhaustive unique probe search for the genome of S. cerevisiae. (a) and (b) are tested on the single 32-bit system; (c) and (d) on the 64-bit system.

Table 3. Optimal jumps for the genome of S. cerevisiae

Short probe length				Long probe length			
	l=20	l=25	l=30		l=40	l=50	l=60
k=2	13	18	22	k=6	23	32	40
k=3	11	15	20	k=8	19	28	37
k=4	9	13	18	k=10	16	24	34
k=5	7	11	15	k=12	12	21	29
k=6	6	9	13	k=14	9	16	23

unique probe search of various genome datasets[6,9,10]. The tests were run on a dual Pentium III 550Mhz system with 256M RAM. The exhaustive search comprised the checking of all 25 nucleotide long probes of a full genome while allowing for up to 4 errors. The running time of the fastest method, i.e. the method using indexed filtering with 1-neighborhood, in the case of the genomes of E. coli and S. cerevisiae were 43.4 hours and 190.6 hours, respectively. The running time of our algorithm for the genomes of E. coli and S. cerevisiae under the same conditions were 5.2 hours and 20.4 hours, respectively. Taking the difference of the computing power used in the tests into account, the results imply that our algorithm is superior to those used by Hyrrö.

We verified the insensitivity of our algorithm to the length of the probes. We tested short (l=20, 25, 30) and long (l=40, 50, 60) sized probes, searching with 2, 3, 4, 5, and 6 errors for the short ones and with 6, 8, 10, 12, and 14 errors for the long ones. The tested dataset is the genome of S. cerevisiae. In computing the edit distance using bit-parallelism, the short sized probes fit in the 32-bit word and the long sized probes fit in the double words of 32-bit computers or the single word of 64-bit computers. Fig. 5 shows the results of the tests; within the limit of the word size of the processor, a longer probe length leads to a shorter running time.

Table 3 shows that the optimal jumps for all of the tested cases are $h \geq \lfloor l/2 \rfloor - k$ for the genome of S. cerevisiae.

5 Conclusion

A good probe should be specific for exactly one gene. We proposed a new approach to determining the specificity of each probe candidate of a full genome. The key idea behind this approach is that once the similarity between a probe candidate and a target gene sequence is computed, by using the result, it is possible to exclude neighboring probe candidates from further consideration if they cannot be matched. Based on our approach, we can reduce the search space without the loss of any significant matches. The presented algorithm is very simple and efficient and a high filtering ratio is attained without any use of index structure. In addition, it is insensitive to the length of the probe.

References

1. R. Baeza-Yates and G. Navarro. Faster Approximate String Matching. *Algorithmica*, **23(2)**:127-158, 1999.
2. L.-L. Cheng, D. Cheung, and S.-M. Yiu. Approximate String Matching in DNA Sequences. *Proc. the 8th International Conference on Database Systems for Advanced Applications*, 303-310, 2003.
3. A. Halperin, A. Buhot, and E. B. Zhulina. On the hybridization isotherms of DNA microarrays: the Langmuir model and its extentions. *Journal of Physics: Condensed Matter*, **18**:S463-S490, 2006

4. Z. He, L. Wu, M. W. Fields, and J. Zhou. Use of Microarrays with Different Probe Size for Monitoring Gene Expression. *Applied and Environmental Microbiology*, **71(9)**:5154-5162, 2005.
5. T. R. Hughes, M. Mao, *et al.*, Expression profiling using microarray fabricated by an ink-jet oligonucleotides synthesizer. *Nature Biotechnology*, **19**:342-347, 2001.
6. H. Hyrrö. On using two-phase filtering in indexed approximate string matching with application to searching unique oligonucleotides. *Proc. String Processing and Information Retrieval (SPIRE 2001)*, 84-95, 2001.
7. H. Hyrrö. Practical Methods for Approximate String Matching. PhD thesis, Department of Computer Sciences, University of Tampere, Finland, December 2003.
8. H. Hyrrö, J, Fredriksson, and G. Navarro. Increased Bit-Parallelism for Approximate and Multiple String Matching. *ACM Journal of Experimental Algorithmics*, **10(2.06)**, 2005.
9. H. Hyrrö, M. Juhola, and M. Vihinen. On approximate string matching of unique oligonucleotides. *Proc. the 10th World Congress on Health and Medical Informatics (Medinfo 2001)*, **10(Pt2)**:960-964, 2001.
10. H. Hyrrö, M. Juhola, and M. Vihinen. Genome-wide selection of unique and valid oligonucleotides. *Nucleic Acids Research*, **33(13)**:e115, 2005.
11. H. Hyrrö and G. Navarro. A Practical Index for Genome Searching. *Proc. String Processing and Information Retrieval (SPIRE 2003)*, 341-349, 2003.
12. F. Li and G. D. Stormo. Selection of optimal DNA oligos for gene expression array, *Bioinformatics*, **17(11)**:1067-1076, 2001
13. T. Majtan, G. Bukovsk, and J. Timko. DNA Microarrays – Techiques and Applications in Microbial Systems. *Folia Microbiologica*, **49(6)**:635-664, 2004.
14. E. Myers. A sublinear algorithm for approximate keyword searching. *Algorithmica*, **12(4/5)**:345-374, 1994.
15. G. Myers. A fast bit-vector algorithm for approximate string matching based on dynamic progamming. *Journal of the ACM*, **46(3)**:395-415, 1999.
16. G. Navarro. A guided tour to approximate string matching. *ACM Computing Surveys*, **33(1)**:31-88, 2001.
17. G. Navarro and R. Baeza-Yates. A hybrid indexing method for approximate string matching. *Journal of Discrete Algorithms*, **1(1)**:205-239, 2000.
18. E. K. Nordberg. YODA: selecting signature oligonucleotides. *Bioinformatics*, **21(8)**:1365-1370, 2005
19. S. Rahmann. Rapid large-scale oligonucleotide selection for microarrays. *Proc. IEEE Computer Society Bioinformatics Conference 2002*, **1**:54-63, 2002.
20. S. Rahmann. Fast and sensitive probe selection for DNA chips using jumps in matching statistics. *Proc. IEEE Computational Systems Bioinformatics*, **2**:57-64, 2003.
21. W. Sung and W. Lee. Fast and accurate probe selection algorithm for large genomes. *Proc. IEEE Computational Systems Bioinformatics*, **2**:65-74, 2003.
22. J. SantaLucia Jr. and D. Hicks D. The Thermodynamics of DNA Structural Motifs, *Annual Review of Biophysics and Biomolecular Structure*, **33**:415-440, 2004.
23. E. Ukkonen. Finding approximate patterns in strings. *Journal of Algorithms*, **6**:132-137, 1985.
24. E. Ukkonen. Algorithms for approximate string matching. *Information and Control*, **64**:100-118, 1985.

Multiple Sequence Local Alignment Using Monte Carlo EM Algorithm

Chengpeng Bi

Children's Mercy Hospitals, Schools of Medicine, Computing and Engineering
University of Missouri, 2401 Gillham Road, Kansas City, MO 64108, USA
cbi@cmh.edu
http://www.cmh.edu

Abstract. The Expectation Maximization (EM) motif-finding algorithm is one of the most popular *de novo* motif discovery methods. However, the EM algorithm largely depends on its initialization and can be easily trapped in local optima. This paper implements a Monte Carlo version of the EM algorithm that performs multiple sequence local alignment to overcome the drawbacks inherent in conventional EM motif-finding algorithms. The newly implemented algorithm is named as Monte Carlo EM Motif Discovery Algorithm (MCEMDA). MCEMDA starts from an initial model, and then it iteratively performs Monte Carlo simulation and parameter update steps until convergence. MCEMDA is compared with other popular motif-finding algorithms using simulated, prokaryotic and eukaryotic motif sequences. Results show that MCEMDA outperforms other algorithms. MCEMDA successfully discovers a helix-turn-helix motif in protein sequences as well. It provides a general framework for motif-finding algorithm development. A website of this program will be available at http://motif.cmh.edu.

Keywords: Expectation Maximization (EM), Monte Carlo EM, Motif Discovery, Multiple Sequence Local Alignment, Transcriptional Regulation.

1 Introduction

A critical task in functional genomics is deciphering the genetic regulatory codes (i.e. motifs). Although many endeavors have been attempted and numerous algorithms have been developed in the past two decades [1], breaking the regulatory codes in a genome remains challenging because motifs are typically short, degenerate, and obey few rules [2]. A particularly successful class of computational methods for the motif discovery problem adopts a position weight matrix (PWM) update approach [3] based on the expectation maximization (EM) algorithms [4]. The EM-based motif discovery algorithm was first developed using PWM-based statistical modeling by Lawrence and Reilly [3]. This methodology has been generalized to one of the most popular motif-finding software packages called MEME [5]. The EM algorithm has been widely deployed in scientific computing due to its simplicity and stability.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 465–476, 2007.

However, despite some appealing features, the EM algorithm has several well-documented limitations: (1) it is strongly dependent on its starting position, and (2) it can provide a saddle point of the likelihood function rather than a local maximum. It has been shown that the Monte Carlo EM (MCEM) algorithm can fix the above limitations to some extent [6]. The MCEM algorithm introduced by Wei and Tanner [7] is a modification of the EM algorithm where the expectation in the E-step is computed numerically through Monte Carlo simulations. It was designed to cope with situations where E-step is hard to compute. More specifically, let the complete data be $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{y})$, here $\boldsymbol{x} \in \mathcal{X}$ is known and $\boldsymbol{y} \in \mathcal{Y}$ unobserved, the t-th step is: (i) Generate iid samples, $\boldsymbol{y}^{(t,1)}$, ..., $\boldsymbol{y}^{(t,j)}$, ..., $\boldsymbol{y}^{(t,m)}$ from $u(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)})$[1] through Monte Carlo simulation, or namely S-step; and (ii) Empirically update the current approximation to the expected log-likelihood function $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)})$[2] through a update or U-step as,

$$Q^{(t+1)}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}) \equiv Q^{(t+1)}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)}; \boldsymbol{y}^{(t,1)}, \ldots, \boldsymbol{y}^{(t,m)}) = \frac{1}{m}\sum_{r=1}^{m} logf(\boldsymbol{x}, \boldsymbol{y}^{(t,r)}|\boldsymbol{\theta}) \quad (1)$$

Finally the M-step determines $\boldsymbol{\theta}^{(t+1)} = argmax_{\theta}\{Q^{(t+1)}(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})\}$. The S- and U-steps iteratively progress until the algorithm converges [7,8]. This paper designs and implements a MCEM algorithm for motif-finding problem. The newly implemented algorithm is named as Monte Carlo EM Motif Discovery Algorithm (MCEMDA). It starts from an initial model, and then iteratively performs Monte Carlo simulation and parameter update steps until convergence. MCEMDA performs very well in the examples presented in the paper.

The rest of this article is organized as follows: Section 2 introduces a framework for motif discovery problems; Section 3 describes the MCEMDA algorithm; Section 4 implements the algorithm; Section 5 gives experimental examples and compares with other EM and Monte Carlo based algorithms; finally Section 6 concludes with discussion.

2 Multiple Local Alignment for Motif Discovery

Let $\mathbf{S} = \{S_1, \ldots, S_i, \ldots, S_N\}$ denote the sequence data set. Let L_i be the length of the sequence i (S_i) and S_{ij} denote a residue symbol taking on a value in K, for instance, $K = \{A, C, G, T\}$. Let $|K|$ be the number of letters in a biological sequence alphabet (i.e., $|K| = 4$ for DNA, and $|K| = 20$ for protein sequences). If only one motif per sequence (i.e. *oops* model) is assumed, there are N motifs in total for N sequences. A zero or one motif per sequence (i.e. *zoops*) model is also frequently used. Nonetheless, both *oops* and *zoops* models assume that

[1] Note that $u(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta})$ is the marginal distribution of the unobserved data $\boldsymbol{y}$ and $\boldsymbol{\theta}$ is the model parameter to be estimated. The complete-data log-likelihood is thus defined as: $L(\boldsymbol{\theta}|\boldsymbol{z}) = logf(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta})$ which cannot be directly solved via the Maximum Likelihood method.

[2] The expected value (or Q-function) of the complete-data log-likelihood in an EM algorithm [4] is defined as, $Q(\boldsymbol{\theta}; \boldsymbol{\theta}^{(t)}) = \int_{\mathcal{Y}} logf(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}) u(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}^{(t)}) d\boldsymbol{y}$.

sequence data come from a two-component multinomial mixture model: (1) the background model, assuming that residues at non-motif positions follow an independent and identical multinomial distribution ($\boldsymbol{\theta}_0$); and (2) the w-mer motif model, assuming that residues within the motif are independent but not identical, in other words, residues at different motif positions come from different multinomial distributions ($\boldsymbol{\theta}_j$). A motif sequence can be thought of drawing from a product of multinomial distributions: $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_j, \ldots, \boldsymbol{\theta}_w]$.

Let A_i be the indicator variable drawing from the location space $\{0, 1\}^{(L_i - w + 1)}$, $\mathbf{A} = [A_1, \ldots, A_i, \ldots, A_N]^T$ be the set of indicator variables representing the motif start sites (i.e. a local alignment) in the sequences, and w be the motif width. The total number of local alignments (V)[3] can be generally expressed as: $V = \prod_{i=1}^{N} \binom{L_i - w + 1}{|A_i|}$, and here the number of motif sites on sequence i is defined as: $|A_i| = \sum_l A_{il}$. Therefore, if $|A_i| = 1$ for all i, it is an *oops* model, otherwise it is a *zoops* or multiple-site model. The total number of motif sites is $|\mathbf{A}| = \sum_i |A_i|$. Alternatively, an indicator variable $a_i = l$ is used to represent the motif starting at position l on sequence i, which is equivalent to $A_{il} = 1$. Note that $a_i = 0$ means no motifs found on sequence i. If multiple sites occur on a sequence, a vector ($\boldsymbol{a}_i$) is used to store all the positions. Obviously a motif indicator vector $\boldsymbol{a}_i \in \mathcal{P}_i(\{1, 2, \ldots, L_i - w + 1\})$, here $\mathcal{P}_i$ is the power set of the i-th sequence motif sites. The alignment of motif sites is initialized by randomly generating a set of motif start sites (i.e. $\mathbf{A}^{(0)}$ or equivalently $[\boldsymbol{a}_1^{(0)}, \ldots, \boldsymbol{a}_N^{(0)}]^T$) and then it is progressively refined until convergence.

Multiple local alignment is the most frequently used method to solve the motif discovery problem. Each alignment can be thought of as a hidden state in the alignment space. The motif discovery problem can therefore be formulated as finding the optimized alignment state (v^*) among the entire alignment space. Index a state by $v \equiv [\boldsymbol{a}_1, \ldots, \boldsymbol{a}_i, \ldots, \boldsymbol{a}_N]^T = \mathbf{A}^{(v)}$, and let the energy of state v be $E(v) = E(\mathbf{S}, \mathbf{A}^{(v)})$ where $\mathbf{A}^{(v)}$ is the alignment corresponding to the state v. The energy may be related to an alignment score or the motif sequence specificity/binding energy [9]. Then at equilibrium the probability of state v (p^v) is proportional to $exp[-E(\mathbf{S}, \mathbf{A}^{(v)})/k_B T]$. Therefore, the optimized alignment state (v^*) is the one with the maximum probability (p^*). If v^* is found, then the parameter estimation ($\boldsymbol{\Theta}^*$) is done. However computing the partition function[4] is intractable, because the alignment problem is *NP*-complete [10].

3 Motif Discovery Using Monte Carlo EM Algorithm

3.1 EM Motif-Finding Algorithms

Since the motif locations ($\mathbf{A}$) are unobserved, the EM algorithm is used to estimate the model parameters given the observed sequences ($\mathbf{S}$). The full data

[3] If it is an *oops*, $V = \prod_{i=1}^{N}(L_i - w + 1)$; if sequence i has no sites at all, then $\binom{L_i}{0} = 1$.

[4] Note the partition function is: $Z = \int_{\mathcal{P}_1} \cdots \int_{\mathcal{P}_N} exp\left\{-E(\mathbf{S}, \mathbf{A}^{(v)})/k_B T\right\} d\boldsymbol{a}_1 \cdots d\boldsymbol{a}_N$. Then, the state probability p^v is simply computed as, $\frac{1}{Z} exp\left(-E(\mathbf{S}, \mathbf{A}^{(v)})/k_B T\right)$.

for motif sequence model is $(\mathbf{S}, \mathbf{A}) = \{(S_i, A_i) : i = \{1, \ldots, N\}\}$. The conditional likelihood of sequence i, given the hidden variables (a_i), is as follows,

$$p(S_i|a_i = l, \boldsymbol{\Theta}, \boldsymbol{\theta}_0) = \prod_{y \in A_{il}^c} \prod_{k \in K} \theta_{0k}^{I(S_{iy}=k)} \prod_{m=1}^{w} \prod_{k \in K} \theta_{mk}^{I(S_{i,l+m-1}=k)} \tag{2}$$

where A_{il}^c denotes the background sites and $I(\cdot)$ is the indicator function. Although equation (2) is for an *oops* model, it is easy to extend to other motif models [11]. To simplify notation, $\boldsymbol{\Theta}$ is used to contain the background parameters $(\boldsymbol{\theta}_0)$ in the following derivation. Let $\boldsymbol{\Theta}^{(t)}$ be the parameter estimates after t-th iteration, then conditional expected complete data log-likelihood given the observed data is often referred to as the Q-function which is defined as,

$$Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)}) = E[ln(p(\mathbf{S}, \mathbf{A}|\boldsymbol{\Theta}))|\mathbf{S}, \boldsymbol{\Theta}^{(t)}] \tag{3}$$

$$= \sum_{i=1}^{N} \sum_{l=1}^{L_i-w+1} p(a_i = l|S_i, \boldsymbol{\Theta}^{(t)}) ln(p(S_i|a_i = l, \boldsymbol{\Theta}))$$

The EM-based motif-finding algorithms maximize the above Q-function by iteratively performing the E- and M-steps [3,11]. The E-step calculates a conditional probability of each potential site,

$$p(a_i = l|S_i, \boldsymbol{\Theta}^{(t)}) = \frac{p(S_i|a_i = l, \boldsymbol{\Theta}^{(t)})}{\sum_{j=1}^{L_i-w+1} p(S_i|a_i = j, \boldsymbol{\Theta}^{(t)})} \tag{4}$$

The M-step computes the updated estimation [3,5,11]. The EM algorithm merely iterates E- and M-steps a number of times until it converges [4].

3.2 Monte Carlo EM Motif-Finding Algorithms

This section describes a special case of the originally reported MCEM algorithm (i.e. one simulation per sequence or $m = 1$) [7] and then it can be easily generalized in the next section. The basic idea underlying MCEMDA is to replace the computation and maximization of $Q(\boldsymbol{\Theta}; \boldsymbol{\Theta}^{(t)})$ by computing $p(a_i = l|S_i, \boldsymbol{\Theta}^{(t)})$ on which an unobserved complete pseudo-sample $\mathbf{A}^{(t)}$ is drawn or aligned (i.e. S-step), followed by an update on $\boldsymbol{\Theta}^{(t+1)}$ (i.e. the U-step). The S- and U-steps iterate until convergence (i.e. a criterion is satisfied). Since the updated parameters lead to a generalized EM (GEM) estimation [4,12], this iterative procedure eventually converges to a better local optimum [6,8]. The following formula is derived [11] as the target density function of the unobserved data (a):

$$u(a_i = l|S_i, \hat{\boldsymbol{\Theta}}^{(t)}) = \frac{\prod_{j=1}^{w} \prod_{k \in K} \left(\frac{\hat{\theta}_{jk}^{(t)}}{\hat{\theta}_{0k}^{(t)}} \right)^{I(S_{i,l+j-1}=k)}}{\sum_{q=1}^{L_i-w+1} \left\{ \prod_{j=1}^{w} \prod_{k \in K} \left(\frac{\hat{\theta}_{jk}^{(t)}}{\hat{\theta}_{0k}^{(t)}} \right)^{I(S_{i,q+j-1}=k)} \right\}} \tag{5}$$

The parameter update procedure is reduced to counting residues as follows,

$$\hat{\theta}_{jk}^{(t)} = \frac{\sum_{i=1}^{N} I(S_{i,a_i+j-1} = k) + \beta_{jk}}{\sum_{i=1}^{N} \sum_{k \in K} I(S_{i,a_i+j-1} = k) + |\boldsymbol{\beta}_j|}$$

where a_i is the motif start position on sequence i, and $|\boldsymbol{\beta}_j| = \sum_k |\beta_{jk}|$. The vector $\boldsymbol{\beta}$ is pseudo-counts or has the meaning of Dirichlet prior distribution in Bayesian formulation. An identical pseudo-count vector is applied to each motif position: $\boldsymbol{\beta}_j = \boldsymbol{\beta}$, where $j \in \{1, \ldots, w\}$. The major advantage of using pseudo-counts is to avoid zero-count of a residue in a motif position due to a small sample size. Here $|\boldsymbol{\beta}| = 1.5$ for DNA sequences, and $|\boldsymbol{\beta}| = sqrt(|\mathbf{A}|)$ for amino acids sequences [13]. Given the current estimate of motif matrix (i.e. $\hat{\boldsymbol{\Theta}}^{(t)}$) and number of motif sites (i.e. $|\mathbf{A}|$), the Q-function for t-th iteration can be evaluated as,

$$Q^{(t)} = |\mathbf{A}| \sum_{k \in K} \hat{\theta}_{0k}^{(t)} ln(\hat{\theta}_{0k}^{(t)}) + |\mathbf{A}| \sum_{j=1}^{w} \sum_{k \in K} \hat{\theta}_{jk}^{(t)} ln(\hat{\theta}_{jk}^{(t)}) \qquad (6)$$

Note that the $Q^{(t)}$-function is a summation of two mixture negative entropy functions times the number of motif sites. The scoring Q-function can be normalized to represent the binding free energy [9].

4 Implementation

Given a set of sequences ($\mathbf{S}$), a motif width (w), maximum simulation (m) and iterations (t_{max}), the MCEMDA algorithm starts via randomly initializing an alignment ($\mathbf{A}^{(0)}$). It then computes the initial model ($\boldsymbol{\Theta}^{(0)}$) and proceeds by simulation (i.e. drawing a new set of sites) and parameter update. The pseudocode of the generalized MCEM motif-finding algorithm is given below,

```
program MCEMDA (S, w, m, t_max)
   initialize motif start positions: A(0)
   evaluate Theta(0), Q(0)
   t := 0, Q_max := Q(0)
   repeat:
     t := t + 1
     for r := 1 to m {
        for i := 1 to N {
           compute probabilities based on Eq. 5
           draw a sample according to the cdf function U
        }
        update Theta(t,r), Q(t,r)
     }
     evaluate Theta(t), Q(t) by averaging Theta(t,r) and Q(t,r)
     if Q(t) > Q_max
        Q_max := Q(t) and Theta* := Theta(t)
   until t := t_max
   output the best alignment & its motif model
end.
```

To achieve efficient computation, one can pre-compute the frequencies of each residue type given a total number of motif sites and their associated logarithm values, and store them in arrays. Note that each motif site is encoded as an object consisting of its start position, associated Q-value, its strand attribute (forward or reverse), and motif probability. While initializing the motif sites in double helix DNA sequences, one can flip a coin to decide whether the motif is on the forward or reverse strand.

To draw an iid sample from a sequence, one ought to compute the cumulative density function (*cdf*), $U(\boldsymbol{a}_i|S_i, \boldsymbol{\Theta}^{(t)})$, conditioning on previous estimate ($\boldsymbol{\Theta}^{(t)}$). New sites ($\boldsymbol{a}_i$) can be determined by repeatedly generating a sequence of random number ($\boldsymbol{c}$) uniformly in $[0, 1]$ such that: $U(\boldsymbol{a}_i - 1|S_i, \boldsymbol{\Theta}^{(t)})) < \boldsymbol{c} \leq U(\boldsymbol{a}_i|S_i, \boldsymbol{\Theta}^{(t)})$. Note that if it is an *oops* model, then a single draw is needed on each sequence. The sampling procedure iterates until a specified number of simulations (m) and iterations (t_{max}). The final output is the best alignment ($\mathbf{A}^*$) found in all rounds and its associated motif matrix ($\boldsymbol{\Theta}^*$). Furthermore, it shall be easy to see that MCEMDA provides a general framework for developing maximum likelihood-based motif discovery algorithms (detailed in the discussion section).

5 Experimental Results

This section describes motif discovery experiments using the MCEMDA algorithm implemented in C++ and other related methods. MCEMDA performs multiple local alignment on either DNA or protein sequences. The MCEMDA algorithm (case $m = 1$ is tested in this report) is compared with other similar algorithms, i.e. EM-based MEME [5] and Gibbs sampling-based BioProspector [14], to demonstrate its robust performance. MEME and BioProspector are two of the most popular motif discovery algorithms. The motif width is chosen in a way that each algorithm achieves its best performance but shall be very close to its known biologically verified length. For other options, MEME and BioProspector defaulted to their own settings. MCEMDA set its maximum iterations of 5,000 per run if not specified otherwise. Their running times were not compared because programs were executed on different machines.

5.1 Motif Discovery in Simulated DNA Sequences

The simulated data sets are generated to test the performance of the MCEMDA algorithm. Eukaryotic sequence motifs are usually highly degenerate and subtle in nature, and therefore the planted motif width is set as 11 base pairs long. Each simulated dataset contains 100 sequences, each of which is 100 nucleotides long. Two simulated datasets are made of either high motif conservation level or low motif conservation level. A high conservation motif is formed such that at any position a dominant nucleotide has a probability of 0.91 and each of the rest positions is 0.03. A low conservation motif is formed such that at any position a dominant nucleotide has a probability of 0.70 and each of the rest is 0.10. A fair Bernoulli coin is tossed in order for each motif sequence to have an equal likelihood of being implanted either on the forward or backward strand. The location

Table 1. Algorithm performance on simulation data*

	High Level			Low Level		
	MCEM	MEME	BioPros	MCEM	MEME	BioPros
B1	**98**	96	40	**68**	44	37
B2	**100**	99	49	**66**	34	47
B3	**99**	**99**	34	**71**	49	32

* The motif width is fixed at 11 base pairs long for all algorithms. The performance is defined as the number of predicted motif sites that are true sites divided by the number of predicted motif sites and then times 100. The number in bold indicates the best performance in each set.

where a motif is implanted is randomly generated. Motifs are planted in three different background sequences (i.e. B1-B3): (1) B1 is uniform (all nucleotide types are equally likely), (2) B2 is AT-rich (AT content = 60% and GC = 40%), and (3) B3 is GC-rich (GC = 60% and AT = 40%).

Table 1 compares the MCEMDA algorithm with MEME and Gibbs motif-finding algorithms using simulated data. It showed that MCEMDA is the best predictor in all backgrounds with different conserved motif sequences. Although MEME and MCEMDA perform equally well in highly conserved motif sequences, MEME is not as good as MCEMDA in low conservation cases. EM algorithms perform greedy searches and thus easily get trapped in a local optimum in low conserved cases. To fix this, MEME incorporates other strategies such as generating smart initial seeds to alleviate the initialization difficulty. In contrast, it is not essential for MCEMDA to bring in other methods to overcome the obstacle.

BioProspector is a Gibbs motif sampler. Its performance is not as good as the other algorithms' in all datasets. In case $m = 1$, MCEMDA is different from BioProspector in that the former updates parameters after each round (i.e. after sampling an alignment or all the motif sequences have been done), whereas BioProspector performs a new update right after sampling each sequence and the current sequence sampling depends on its previous single sequence update. The strategy used in BioProspector may give rise to very quick convergence, and thus most likely predispose the sampler to local optima.

5.2 Motif Discovery in Real Biological Sequences

Five annotated motif datasets are used: three of them from bacterial genomes (i.e. CRP, FNR and LexA) and two from eukaryotic TF binding sites (ERE and E2F). Three bacterial TF binding sites (i.e. CRP, FNR and LexA) share the same sequence property: all have two core sub-motifs interrupted by low conserved nucleotides (gap spacer) in between (see Fig. 1A-1C). The presence of the spacer makes the final motif model more degenerate and influences the performance of the motif-finding algorithms. The CRP dataset is taken from the previous papers [3,11,14]. There are one or two known CRP-binding sites in each of the 18 sequences in the dataset, and the location of these binding sites has been well documented. Each determined binding site motif length is 22 base

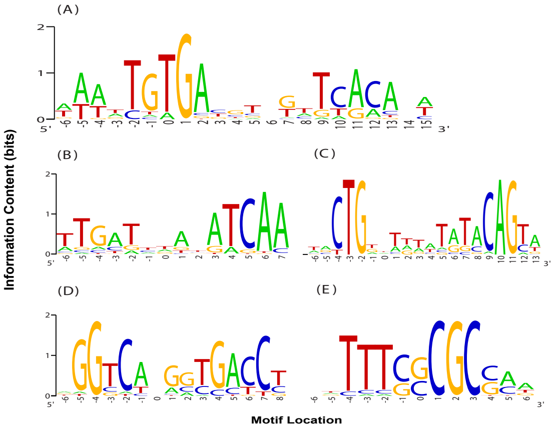


Fig. 1. Sequence logos of five annotated transcription factor binding sites. The sequence logos [15] are plotted using WebLogo [16]. (A) CRP binding sites (18 sequences); (B) FNR protein binding sites (67 sequences); (C) LexA binding sites (24 sequences); (D) ERE binding elements (25 sequences); (E) E2F binding site sequences (25 sequences).

pairs. However, MEME achieved its best performance in 24-mer motif model (Table 2).

Apart from the CRP dataset, two other annotated bacterial motif datasets (i.e. the FNR and LexA TF binding sites) are also extracted from the RegulonDB [17]. The FNR motif is described in the literature as an interrupted 14-bp palindrome and the consensus, TTGACnnnnATCAA, consists of two conserved 5-bp blocks separated by a fixed 4-bp spacer. The LexA motif length is 20 base pairs long with two core motifs on the ends and it is interrupted by low-conserved nucleotides in between as well. Two eukaryotic motif datasets (ERE and E2F) contain short motif sequences extracted from references [18,19]. The estrogen receptor (ER), an important nuclear hormone receptor, is a ligand-induced enhancer protein bound with the estrogen response elements (EREs), and it regulates downstream genes. There are 25 ERE genomic sequences, each 200 base pairs long and its motif length is known to be 15 base pairs long. The E2F transcription factor [18] binding dataset contains 25 genomic sequences, each 200 base pairs long with 27 embedded motif sites.

Table 2 summarizes the motif algorithm performance using the F-score [20]. Different motif widths are used in a way that algorithms can achieve their best performance. However, MCEMDA chose the known motif width in all cases. The MCEMDA algorithm is the best predictor (F-score is 0.912) compared to others. MEME has better average performance (0.776) than BioProspectors (0.602), but is not as good as MCEMDA in tested samples. Figure 2 shows the sequence logos for all the five transcription factor binding motifs built from the MCEMDA algorithm. The CRP motif consists of two core sub-motifs (5-mer) separated by six unspecified nucleotides: TGTGAnnnnnnTCACT, which is congruous with biological findings. The FNR and LexA motifs have consensus

Table 2. Algorithm comparison on biological real data*

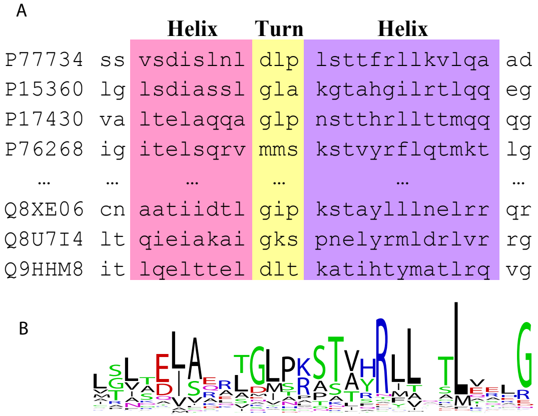
TF	Algorithms	width(w)	sites($\|\mathbf{A}\|$)	Precision	Recall	F-score
CRP	MCEMDA	22	18	18/18	18/23	0.878
	BioPros	22	9	9/9	9/23	0.563
	MEME	24	13	12/13	12/23	0.667
FNR	MCEMDA	14	67	64/67	64/67	0.955
	BioPros	14	67	39/67	39/67	0.575
	MEME	20	43	43/43	43/67	0.782
LexA	MCEMDA	20	24	24/24	24/24	1.000
	BioPros	20	24	15/24	15/24	0.625
	MEME	21	24	23/24	23/24	0.958
ERE	MCEMDA	15	25	22/25	22/25	0.880
	BioPros	13	16	14/16	14/25	0.683
	MEME	15	17	15/17	15/25	0.714
E2F	MCEMDA	13	25	22/25	22/27	0.846
	BioPros	11	21	11/21	11/27	0.564
	MEME	13	23	19/23	19/27	0.760

* Note that the results of CRP, ERE and E2F for BioProspector and MEME algorithms are obtained from [20]. The precision is defined as the number of predicted motif sites that are true sites divided by the number of predicted motif sites [20]. The recall is the number of predicted motif sites that are true sites divided by the number of true sites. The F-score is computed as: F = 2 * precision*recall / (precision + recall).

sequences of TTGATnnnnATCAA and nnCTGnnwwwwwrwmCAGyd, respectively, and these are in line with their validated motif sites. Two short motifs (ERE and E2F) are slightly higher conserved, and the identified consensus sequences are in agreement with the annotated motif sequences [18,19].

5.3 Motif Discovery in Protein Sequences

The helix-turn-helix (HTH) is composed of two almost perpendicular α helices linked by a several-residue β turn. The HTH motif is a common recognition element used by transcription regulators of prokaryotes and eukaryotes. Many bacterial transcription regulators which bind DNA through a HTH motif can be classified into subfamilies on the basis of sequence similarities. *IclR* is one of the subfamilies, and it is the repressor of the acetate operon in bacteria. There are 352 non-redundant *IclR* protein sequences downloaded from the BacTregulators website [21]. The sequence length ranges from 181 to 845 residues. The HTH DNA binding domain is known to contain 24 amino acids as highlighted in Figure 2A. MCEMDA was run five times each with 500 iterations to perform the protein sequence alignment. The motif length is set to 28 residues including 2 flanking residues. The HTH motif sites in the *IclR* protein sequences are successfully detected and the associated motif model is built as illustrated in the protein sequence logo (Figure 2B), which conforms to the annotated



		Helix	Turn	Helix	
P77734	ss	vsdislnl	dlp	lsttfrllkvlqa	ad
P15360	lg	lsdiassl	gla	kgtahgilrtlqq	eg
P17430	va	ltelaqqa	glp	nstthrllttmqq	qg
P76268	ig	itelsqrv	mms	kstvyrflqtmkt	lg
…	…	…	…	…	…
Q8XE06	cn	aatiidtl	gip	kstaylllnelrr	qr
Q8U7I4	lt	qieiakai	gks	pnelyrmldrlvr	rg
Q9HHM8	it	lqelttel	dlt	katihtymatlrq	vg

Fig. 2. HTH motif alignment and its sequence logo. (A) Multiple local alignment of 352 nonredundant protein sequences (shown in part), the highlighted parts from left to right are α helix (α_2, 8 residues), β turn (3 residues) and α helix (α_3, 13 residues) respectively; (B) Sequence logo [15] plotted using WebLogo [16].

documentation [22]. Although the example shows the new algorithm's capability of performing large-scale protein sequence alignment, the algorithm shall be extended to simultaneously discover several distinct protein motifs in the future.

6 Discussion

The MCEMDA algorithm to some extent fixes the problems inherent in EM motif algorithms due to its stochastic properties. Results showed its robust performance since MCEMDA increases the chance of escaping from inferior local optima and does not depend on its initialization given a long period of iterations. In contrast to MCEMDA, MEME integrates other methods such as smart seed initialization in order to achieve a better convergence [5]. Algorithm performance comparison using simulated and real motif sequence datasets shows that the sampling strategy in MCEMDA is better than that used in BioProspector. The Gibbs sampling tactics employed in BioProspector contribute to its expeditious convergence, whereas it would most likely give rise to a local optimum.

This paper presents a general framework for developing motif-finding algorithms based on the maximum log-likelihood model. First, if the simulation size (m) equals to 1, MCEMDA can be thought of as a simple Markov Chain Monte Carlo algorithm, or a special case of Metropolis-Hastings algorithms [11] with accept rate $\alpha_M = 1.0$, i.e. each Monte Carlo move is always accepted. The Markov chain is evolved such that Monte Carlo sampling is based on the current estimation of parameter vector and its random walk explores as much alignment space as possible. MCEMDA keeps track of the best solution along the Monte Carlo iterative sampling process. Second, if its parameter update step is performed right after drawing from each sequence, then MCEMDA is reduced to the Gibbs motif sampler [13,14]. Last, if m equals to the number of all the potential motif sites in the sequence data set, then MCEMDA is the same as ordinary EM motif-finding algorithms [3,5]. It would be intriguing to design a synergistic scheme that can take advantages of each above strategy in the near future.

Moreover, the current algorithm shall be extended to tackle more general motif-finding issues, i.e. discover multiple sites per sequence or several distinct motifs in one run. Motif width is usually unknown. The maximum log-likelihood model can be generalized to the minimum description length (MDL) principle whereby the motif width can be assimilated into the total code length. Therefore one would argue that MDL criterion might be better utilized to deal with w-mer motif model selection [23].

Currently large-scale multiple alignment of motif sequences is common in bioinformatics, and using parallel processing machines such as cluster computing facilities, MCEMDA can send many runs each of which has a sufficiently long Markov chain to cluster nodes, and then collect solutions from nodes and rank them to output the top best models. This simple parallel algorithm would remarkably ameliorate the dilemma of whether a single long chain or multiple short chains shall be applied to the Monte Carlo simulation.

Acknowledgments. The author would like to thank Heather L. Newkirk for text editing, and Zhi Wei for providing eukaryotic motif sequences. This research is supported in part by the Katharine B. Richardson Foundation.

References

1. MacIsaac, K.D. and Fraenkel, E.: Practical Strategies for Discovering Regulatory DNA Sequence Motifs. PLoS Comput Biol, **2** (2006) e36
2. Tompa, M. et al.: Assessing Computational Tools for the Discovery of Transcription Factor Binding Sites. Nature Biotechnology **23** (2005) 137-144
3. Lawrence, C.E. and Reilly, A.A.: An Expectation Maximization Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences. Proteins: Structure, Function and Genetics **7** (1990) 41-51
4. Dempster, A.P. et al.: Maximum Likelihood from Incomplete Data via the EM Algorithm (with Discussion). J. the Royal Statist. Soc. B **39** (1977) 1-38
5. Bailey, T.L. and Elkan, C.: Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization. Machine Learning **21** (1995) 51-80
6. Celeux, G et al.: Stochastic Versions of the EM Algorithm: An Experimental Study in the Mixture Case. J. Statist. Comput. Simul. **55** (1996) 287-314
7. Wei, G.C.G. and Tanner, M.A.: A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms. Journal of the American Statistical Association **85** (1990) 699-704
8. Delyon, B. et al.: Convergence of a Stochastic Approximation Version of the EM Algorithm. Ann. Statist. **27** (1999) 94-128
9. Berg, O.G. and von Hippel, P.H.: Selection of DNA Binding Sites by Regulatory Proteins: Statistical-mechanical Theory and Application to Operators and Promoters. Journal of Molecular Biology **193** (1987) 723-750
10. Bonizzoni, P. and Vedova, G.D.: The Complexity of Multiple Sequence Alignment with SP-score That Is a Metric. Theoretical Computer Science **259** (2001) 6379

11. Bi, C.-P.: SEAM: A Stochastic EM-type Algorithm for Motif-Finding in Biopolymer Sequences. J. Bioinformatics and Comput. Biol. (2007) in press
12. Wu, C.F.J.: On the Convergence Properties of the EM Algorithm. The Annals of Statistics **11** (1983) 95-103
13. Lawrence, C.E. et al.: Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment. Science **262** (1993) 208-214
14. Liu, X. et al.: BioProspector: Discovering Conserved DNA Motifs in Upstream Regulatory Regions of Co-expressed Genes. Pacific Symposium on Biocomputing **6** (2001) 127-138
15. Schneider, T.D. and Stephens, R.M.: Sequence Logos: A New Way to Display Consensus Sequences. Nucleic Acids Research **18** (1990) 6097-6100.
16. Crooks, G.E. et al.: WebLogo: A Sequence Logo Generator. Genome Research **14** (2004) 1188-1190
17. Salgado, H. et al.: RegulonDB (version 5.0): Escherichia coli K-12 Transcriptional Regulatory Network, Operon Organization, and Growth Conditions. Nucleic Acids Res. **34** (2006) D394-7
18. Kel, A.E. et al.: Computer-assisted Identification of Cell Cycle-related Genes: New Targets for E2F Transcription Factors. J. Mol. Biol. **309** (2001) 99-120
19. Klinge, C.M.: Estrogen Receptor Interaction with Estrogen Response Elements. Nucleic Acids Res. **29** (2001) 2905-2919
20. Wei, Z. and Jensen, S.T.: GAME: Detecting cis-Regulatory Elements Using a Genetic Algorithm. Bioinformatics **22** (2006) 1577-1584
21. Martnez-Bueno1, M. et al.: BacTregulators: A Database of Transcriptional Regulators in Bacteria and Archaea. Bioinformatics **20** (2004) 2787-2791
22. Krell, T. et al.: The IclR Family of Transcriptional Activators and Repressors Can Be Defined by a Single Profile. Protein Science **15** (2006) 1207-1213
23. Bi, C.-P.: A Genetic-Based EM Motif-Finding Algorithm for Biological Sequence Analysis. Proceeding of IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (2007) in press

Cancer Class Discovery Using Non-negative Matrix Factorization Based on Alternating Non-negativity-Constrained Least Squares

Hyunsoo Kim and Haesun Park

College of Computing, Georgia Institute of Technology,
266 Ferst Drive, Atlanta, GA 30332-0765
{hskim,hpark}@cc.gatech.edu

Abstract. Many bioinformatics problems deal with chemical concentrations that should be non-negative. Non-negative matrix factorization (NMF) is an approach to take advantage of non-negativity in data. We have recently developed sparse NMF algorithms via alternating non-negativity-constrained least squares in order to obtain sparser basis vectors or sparser mixing coefficients for each sample, which lead to easier interpretation. However, the additional sparsity constraints are not always required. In this paper, we conduct cancer class discovery using NMF based on alternating non-negativity-constrained least squares (NMF/ANLS) without any additional sparsity constraints after introducing a rigorous convergence criterion for biological data analysis.

Keywords: Cancer Class Discovery, Non-negative Matrix Factorization, Convergence Criterion, Non-negativity-constrained Least Squares.

1 Introduction

Since non-negative matrix factorization (NMF) may give us simple interpretation due to non-subtractive combinations of non-negative basis vectors, it has recently received much attention and it has been applied to text data mining [1,2], gene expression data analysis [3,4,5], and so forth. Given a non-negative matrix A of size $m \times n$, where each column of A corresponds to a data point in the m-dimensional space, and a desired rank $k < \min\{m, n\}$, NMF ($A \approx WH \;\; s.t. \;\; W, H \geq 0$) can be obtained by solving the following optimization problem:

$$\min_{W,H} f(W,H) \equiv \frac{1}{2}\|A - WH\|_F^2, \quad s.t. \;\; W, H \geq 0, \tag{1}$$

where $W \in \mathbb{R}^{m \times k}$ is a basis matrix, $H \in \mathbb{R}^{k \times n}$ is a coefficient matrix, and W, $H \geq 0$ means that all elements of W and H are non-negative. Imposing additional sparsity constraints on W or H may give us simpler interpretation. Thus, we have recently developed sparse NMF algorithms via alternating non-negativity-constrained least squares [6]. However, the additional sparsity constraints are not always required unless one is interested in easier interpretation

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 477–487, 2007.

in spite of some possible information loss. In this paper, we focus on NMF algorithms that do not have additional sparsity constraints.

Cancer class discovery using NMF based on multiplicative updating rules [7] has been studied [4]. This work used a convergence criterion using connectivity matrix [4] that can only consider the analytical convergence of H. Since we need to use W for biological analysis in order to investigate the contribution of a gene to each biological process, it is also necessary to guarantee the analytical convergence of W [6]. In this paper, we perform cancer class discovery by NMF based on alternating non-negativity-constrained least squares (NMF/ANLS), which is fast due to its sound convergence property. The rest of this paper is organized as follows. In Section 2, we describe NMF/ANLS and a rigorous convergence criterion. Section 3 presents experimental results illustrating properties of NMF/ANLS. Summary is given in Section 4.

2 NMF Based on Alternating Non-negativity-Constrained Least Squares (NMF/ANLS)

In this section, we describe NMF based on alternating non-negativity-constrained least squares. Paatero and Tapper [8] originally proposed using a constrained alternating least squares algorithm to solve Eq. (1). However, this approach has not been widely used for many pattern recognition applications since an NMF algorithm using 'lsqnonneg' function in Matlab takes too long computing time [9,10]. To enhance computing speed, we utilize the recent achievements on fast algorithms for non-negativity-constrained least squares (NLS). Bro and de Jong [11] made a substantial speed improvement to Lawson and Hanson's algorithm [12] for large scale NLS problems. Van Benthem and Keenan [13] devised an algorithm that further improves the performance of NLS. This algorithm deals with the following NLS optimization problem given $B \in \mathbb{R}^{m\times k}$ and $A \in \mathbb{R}^{m\times n}$:

$$\min_G \|BG - A\|_F^2, \quad s.t. \;\; G \geq 0,$$

where $G \in \mathbb{R}^{k\times n}$ is a solution. It is based on the active/passive set method. More detailed explanations of this algorithm can be found in [13].

Given a non-negative matrix $A \in \mathbb{R}^{m\times n}$, NMF/ANLS starts with the initialization of $H \in \mathbb{R}^{k\times n}$ with non-negative values. Then, it iterates the following ANLS until convergence:

$$\min_W \|H^T W^T - A^T\|_F^2, \quad s.t. \;\; W \geq 0, \tag{2}$$

which fixes H and solves the optimization with respect to W, and

$$\min_H \|WH - A\|_F^2, \quad s.t. \;\; H \geq 0, \tag{3}$$

which fixes W and solves the optimization with respect to H. Alternatively, after initializing W, one can iterate Eq. (3) and Eq. (2) until convergence. We utilize the fastest NLS algorithm [13] to solve Eqs. (2-3). Lin [9] discussed the convergence property of alternating non-negativity-constrained least squares and showed that any limit point of the sequence (W,H) generated by alternating non-negativity-constrained least squares is a stationary point of Eq. (1), and proposed projected gradient methods that do not use lsqnonneg function. To alleviate the uniqueness problem, after convergence, the columns of the basis matrix W are normalized to unit L_2-norm and the rows of H are adjusted so that the approximation error is not changed.

2.1 Analytic Convergence Criterion

Once we have a non-negative decomposition ($A \approx WH \quad s.t. \quad W, H \geq 0$), we can use the basis matrix W to divide the m genes into k gene-clusters and the coefficient matrix H to divide the n samples into k sample-clusters. Typically, gene i is assigned to gene-cluster q if the $W(i,q)$ is the largest element in $W(i,:)$ and sample j is assigned to sample-cluster q if the $H(q,j)$ is the largest element in $H(:,j)$. We test convergence at every five iterations by using these positions of the largest elements in rows of W and columns of H. We assume that NMFs are converged if both the positions of the largest elements in rows of W and the positions of the largest elements in columns of H have not changed during 10 convergence tests. This analytic convergence criterion was proposed in [6].

2.2 Mathematical Convergence Criterion

Here, we describe a rigorous mathematical convergence criterion, which checks if NMF is converged to (near) a stationary point. Most non-convex optimization algorithms guarantee only the stationarity of limit points, which may or may not be a local minimum. The Karush-Kuhn-Tucker (KKT) optimality condition can be used for convergence test. The KKT conditions of NMF problem Eq. (1) are

$$\begin{array}{l} \text{(C1) } W_{iq} \geq 0, \\ \text{(C2) } H_{qj} \geq 0, \\ \text{(C3) } (\partial f(W,H)/\partial W)_{iq} \geq 0, \\ \text{(C4) } (\partial f(W,H)/\partial H)_{qj} \geq 0, \\ \text{(C5) } W_{iq} \cdot (\partial f(W,H)/\partial W)_{iq} = 0, \\ \text{(C6) } H_{qj} \cdot (\partial f(W,H)/\partial H)_{qj} = 0, \forall i,q,j. \end{array} \tag{4}$$

These conditions can be rewritten as

$$\begin{array}{l} \min(W, WHH^T - AH^T) = 0, \\ \min(H, W^TWH - W^TA) = 0, \end{array} \tag{5}$$

where the minimum is taken component wise [14]. Let Δ_o be the KKT residual measured by the L_1-vector norm,

$$\begin{array}{l} \Delta_o = \sum_{i=1}^{m} \sum_{q=1}^{k} |\min(W_{iq}, (WHH^T - AH^T)_{iq})| + \\ \sum_{q=1}^{k} \sum_{j=1}^{n} |\min(H_{qj}, (W^TWH - W^TA)_{qj})|. \end{array} \tag{6}$$

We count the number of W elements that did not converge yet, *i.e.* $\delta_W = \#(\min(W, WHH^T - AH^T) \neq 0)$, and the number of H elements that did not converge yet, *i.e.* $\delta_H = \#(\min(H, W^TWH - W^TA) \neq 0)$. Then, Δ_o is normalized and we defined the following normalized KKT residual:

$$\Delta = \frac{\Delta_o}{\delta_W + \delta_H}, \tag{7}$$

which reflects the average of convergence errors for elements in W and H that did not converge. The mathematical convergence criterion is defined as

$$\Delta \leq \epsilon \Delta_1, \tag{8}$$

where Δ_1 is the Δ value in the first iteration and ϵ is a tolerance.

2.3 Combined Convergence Criterion

In the mathematical convergence criterion, one could set ϵ to 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, or even a smaller value. The smaller ϵ yields the larger number of iterations for convergence, which results in the larger computing time. When $\epsilon = 0$, the KKT conditions are completely satisfied since the mathematical convergence criterion is $\Delta = 0$. However, it is too harsh to use $\epsilon = 0$ in general. In order to reduce computing time, one may use a larger ϵ value. Then, a NMF solution may be farther from a stationary point and may sometimes not satisfy the analytical convergence criterion. Thus, we suggest a combined convergence criterion that requires not only the mathematical convergence but also the analytical convergence that is very important for biological analysis.

3 Experiments and Discussion

3.1 Datasets Description

We used the leukemia gene expression dataset (ALLAML) and the central nervous system tumors dataset (CNS) [4], which are the same datasets used in [6]. The ALLAML dataset contains acute lymphoblastic leukemia (ALL) that has B and T cell subtypes, and acute myelogenous leukemia (AML) that occurs more commonly in adults than in children. This gene expression dataset consists of 38 bone marrow samples (19 ALL-B, 8 ALL-T, and 11 AML) with 5,000 genes. The central nervous system dataset is composed of four categories of CNS tumors with 5,597 genes. It consists of 34 samples representing four distinct morphologies: 10 classic medulloblastomas, 10 malignant gliomas, 10 rhabdoids, and 4 normals. All datasets we used contain only non-negative entries. To make a non-negative matrix for the CNS dataset, a cutoff lower threshold value (20) and an upper threshold value (16,000) were used, as in the previous studies [4,5]. We implemented algorithms in Matlab 6.5 [15]. All our experiments were performed on a P3 600MHz machine with 512MB memory.

Table 1. Performance comparison between NMF using norm-based updating rules (NMF/NUR) [7] and NMF/ANLS on the leukemia dataset with $k = 3$. After 30 runs with different random initializations, we obtained the average values of computing time, percentages of zero elements in W and H, and the number of iterations. Purity and entropy were computing from H that produced the lowest approximation error. *The average percentages of the number of very small non-negative elements that are smaller than 10^{-8} in W and H computed from NMF/NUR.

Algorithms	NMF/NUR	NMF/ANLS
#($W = 0$) (%)	2.72%*	2.71%
#($H = 0$) (%)	17.28%*	18.42%
Purity	0.974	0.974
Entropy	0.095	0.095
# of iterations	3806	91.5
Computing time	159.2 sec.	7.1 sec.

3.2 Clustering Performance Measures

To measure the clustering performance, we used purity and entropy. Suppose we are given l categories (true class labels), while NMF generates k clusters. Purity is given by

$$\text{Purity} = \sum_{q=1}^{k} \frac{n_q}{n} P(\tilde{\Omega}_q), \quad P(\tilde{\Omega}_q) = \frac{1}{n_q} \max_j (n_q^j), \tag{9}$$

where $\tilde{\Omega}_q$ is a particular cluster of size n_q, n_q^j is the number of samples in $\tilde{\Omega}_q$ that belong to original class Ω_j $(1 \le \Omega_j \le l)$, k is the number of clusters, and n is the total number of samples. The larger values of purity, the better clustering performance. Entropy is defined as follows:

$$\text{Entropy} = \sum_{q=1}^{k} \frac{n_q}{n} E(\tilde{\Omega}_q), \quad E(\tilde{\Omega}_q) = -\frac{1}{\log_2 l} \sum_{j=1}^{l} \frac{n_q^j}{n_q} \log_2 \frac{n_q^j}{n_q}, \tag{10}$$

where l denotes the number of original class labels. The smaller values of entropy, the better clustering quality.

3.3 Clustering Performance Comparison

We used the NMF using norm-based updating rules (NMF/NUR) that minimizes the Euclidean distance $\|A - WH\|_F$ in order to compare with NMF/ANLS. We used the combined convergence criterion with $\epsilon = 10^{-5}$. We also imposed a maximal number of 20,000 iterations on each method. Table 1 shows the performance comparison between NMF/NUR and NMF/ANLS on the leukemia data matrix

Table 2. Performance comparison between NMF/NUR [7] and NMF/ANLS on the CNS tumors dataset. After 30 runs with different random initializations, we obtained the average values of computing time (in seconds), percentages of zero elements in W and H, and the number of iterations. *The average percentages of the number of very small non-negative elements that are smaller than 10^{-8} in W and H computed from NMF/NUR.

Algorithm	NMF/NUR		
k	3	4	5
$\#(W=0)$ (%)	8.77%*	9.07%*	12.60%*
$\#(H=0)$ (%)	16.99%*	24.14%*	25.43%*
# of iterations	11151	13770	16717
Computing time	563.5 sec.	836.4 sec.	1334.9 sec.
Algorithm	NMF/ANLS		
k	3	4	5
$\#(W=0)$ (%)	8.69%	9.03%	12.54%
$\#(H=0)$ (%)	18.63%	25.00%	26.88%
# of iterations	105.2	100.3	130.5
Computing time	9.8 sec.	12.1 sec.	20.3 sec.

with $k = 3$. After 30 runs with different random initializations, we computed the average values of computing time, percentage of zero elements in W and H, approximation error, and the number of iterations. Purity and entropy were computing from H that produced the lowest approximation error. It is interesting that NMF/NUR also produced purity=0.974 with only one misclustering, whereas the previous study [4] showed that NMF using divergence-based multiplicative updating rules misclustered two samples. This result may be attributed to the rigorous combined convergence criterion. However, NMF/NUR took much longer computing time than NMF/ANLS. The fast convergence of NMF/ANLS is due to its sound convergence property. NMF/NUR produced very small non-negative elements ($< 10^{-8}$) in W and H, while NMF/ANLS generated the exact zero elements. This is also a very interesting property of NMF/ANLS since exact zero values are helpful in reducing computing complexity and storage requirement for handling sparse datasets.

Table 2 shows the performance comparison on the CNS tumors dataset with various k values. NMF/ANLS was an order of magnitude faster than NMF/NUR. For some of random initializations, NMF/NUR could not converge to a stationary point in terms of the combined convergence criterion within 20,000 iterations. This is a serious problem of NMF/NUR since it may not be practically applicable to huge data analysis due to its slow convergence. Using both NMF/NUR and NMF/ANLS with $k = 4$, we obtained purity=0.971 and entropy=0.071, which means excellent clustering power since there was only one misclustering.

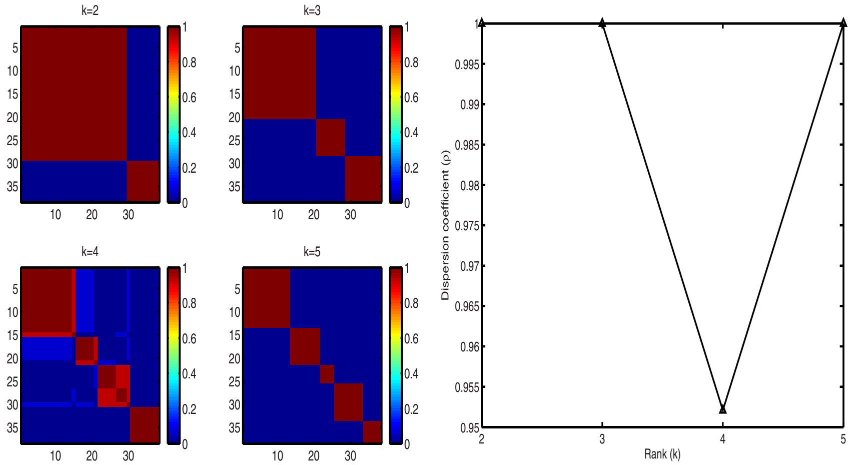


Fig. 1. Leukemia clustering by NMF/ANLS. (Left) The reordered consensus matrices on the leukemia dataset. (Right) The corresponding dispersion coefficients.

3.4 Model Selection

For the determination of the number of clusters in the leukemia dataset and the CNS tumors dataset, we repeated non-negative matrix factorizations 30 times to obtain the average connectivity matrix (*i.e.* consensus matrix) whose entries reflect the probability that samples i and j belong to the same cluster. To measure the dispersion of the consensus matrix C, we defined the dispersion coefficient (ρ) as

$$\rho = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} 4(C_{ij} - \frac{1}{2})^2, \tag{11}$$

of which value is $\rho = 1$ for a perfect consensus matrix (all entries $= 0$ or 1) and $0 \leq \rho < 1$ for a scattered consensus matrix. After obtaining ρ_k values for various k, we can determine the number of clusters from the maximal k producing maximal ρ_k. Figure 1 shows that NMF/ANLS could find the number of clusters in the leukemia dataset due to the maximal ρ_k at $k = 3$. Figure 2 and Figure 3 illustrate that NMFs could find the number of clusters in the CNS tumors dataset due to the maximal ρ_k at $k = 4$, since they generated perfect consensus matrices for $k = 2, 3, 4$. In other words, they produced H matrices that have the same cluster structure with different random initializations of H.

3.5 Biological Analysis

Figure 4 presents the matrices W and H obtained from NMF/ANLS, which produced the lowest approximation error $\|A - WH\|_F$ after 30 runs with different

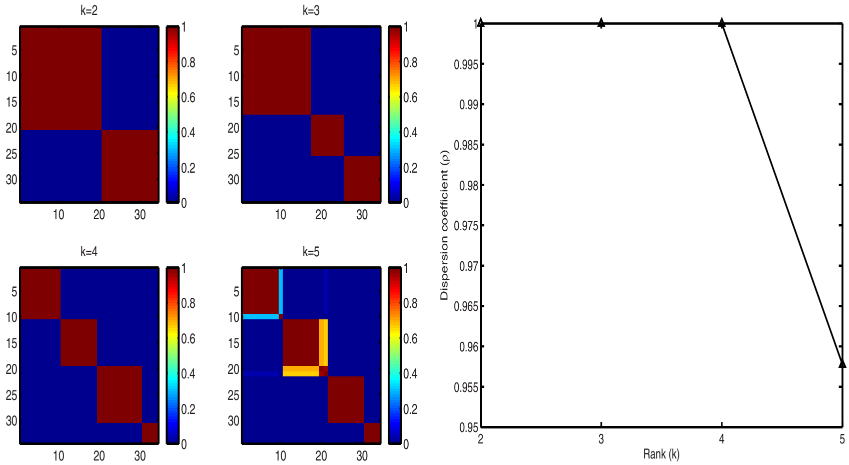


Fig. 2. CNS tumors clustering by NMF using norm-based update rules [7]. (Left) The reordered consensus matrices on the CNS tumors dataset. (Right) The corresponding dispersion coefficients.

random initializations of H. A row vector of the basis matrix W has the contributions of a gene to the k biological pathways or processes (*i.e.* k columns of W). A gene can participate in more than one biological process. One can investigate genes that have relatively large coefficient in each biological process. A column vector of the coefficient matrix H has the contributions of k biological processes to the gene expression of a sample. From the matrix H, we can recognize that ALL-B is dominated by the first biological process. ALL-T is almost controlled by the second biological process. The third biological process is the major component for AML. Some genes dominantly contribute to only single biological pathway or process. For instance, MB-1 gene (U05259) is most active in the first process. Transcription factor 7 (T-cell specific, HMG-box) (TCF7, X59871) is active in the second process, which is also known as T cell factor-1 (TCF-1). Some genes play a major role in the third process, for example, Interleukin 8 (IL8, M28130), DF D component of complement (adipsin) (CFD, M84526), Cystatin C (amyloid angiopathy and cerebral hemorrhage (CST3, M27891), galectin 3 (LGALS3,M57710), Chemokine (C-X-C motif) ligand 2 (CXCL2, M57731), *etc.* Chemokine is a type of cytokines that bind to a specific cell-surface receptor and are critical to the functioning of both innate and adaptive immune responses. Among above genes, MB-1, IL8, CFD, CST3, LGALS3 were the same gene as those found in [16]. Ribosomal protein S3 (RPS3, X57351) simultaneously participates in all three processes. This gene encodes a ribosomal protein that is a component of the 40S subunit, where it forms part of the domain where translation is initiated. It is reasonable since RPS3 is a housekeeping gene and ribosomal protein genes are usually overexpressed in some cancers. We have shown that NMF/ANLS can be used for cancer class discovery and biological process analysis.

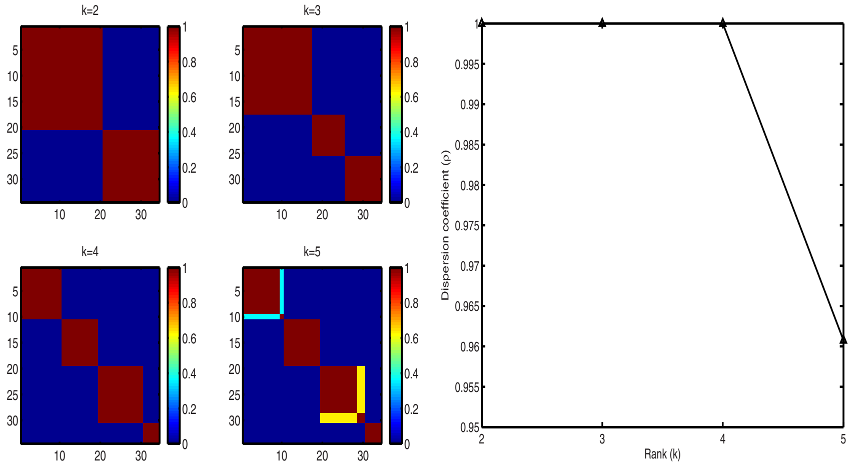


Fig. 3. CNS tumors clustering by NMF/ANLS. (Left) The reordered consensus matrices on the CNS dataset. (Right) The corresponding dispersion coefficients. The dispersion coefficient drops when k increases from 4 to 5, indicating a four-cluster split of the data is more stable than a five-cluster split.

3.6 Remarks on Clustering Via NMF

The object function of NMF $\|A - WH\|_F^2$ can be rewritten as

$$J_{NMF} = \sum_{j=1}^{n} \left\| A(:,j) - \sum_{q=1}^{k} W(:,q) * H(q,j) \right\|_2^2 . \tag{12}$$

The j-th sample $A(:,j)$ is represented as a linear combination of basis vectors. The sample j is assigned to sample-cluster $\tilde{\Omega}_q$ when the q-th coefficient $H(q,j)$ is maximal in $H(:,j)$, in other words, when the q-th basis vector $W(:,q)$ is a dominant component of the sample j. The K-means clustering minimizes

$$J_{K-MEANS} = \sum_{q=1}^{k} \sum_{j \in \mathcal{S}_q} \|A(:,j) - \mathbf{c}_q\|_2^2, \tag{13}$$

where $\mathcal{S}_q$ is a set of column indices of A belonging to cluster $\tilde{\Omega}_q$ and $\mathbf{c}_q$ is the centroid vector of the q-th cluster, which satisfies $\mathbf{c}_q = \text{argmin}_{\mathbf{z}} \sum_{j \in \mathcal{S}_q} \|A(:,j) - \mathbf{z}\|_2^2$. A special NMF $A_+ \approx C_+ \Delta_+$ *s.t.* $A_+, C_+, \Delta_+ \geq 0$, where $C_+ \in \mathbb{R}^{m \times k}$ is a centroid matrix and $\Delta_+ \in \mathbb{R}^{k \times n}$ is a cluster indicator matrix ($(\Delta_+)_{qj} = 1$ if $j \in \mathcal{S}_q$ and $(\Delta_+)_{qj} = 0$ otherwise, $\sum_q (\Delta_+)_{qj} = 1$), is equivalent to K-means [17]. Moreover, a special one-sided NMF $A_\pm \approx C_\pm \Delta_+$ *s.t.* $\Delta_+ \geq 0$ without non-negativity constraints on A and C, where $C_\pm \in \mathbb{R}^{m \times k}$ is a centroid matrix of $A_\pm$, is equivalent to K-means. In these cases, centroid vectors become basis vectors and the j-th sample is represented by only one basis vector.

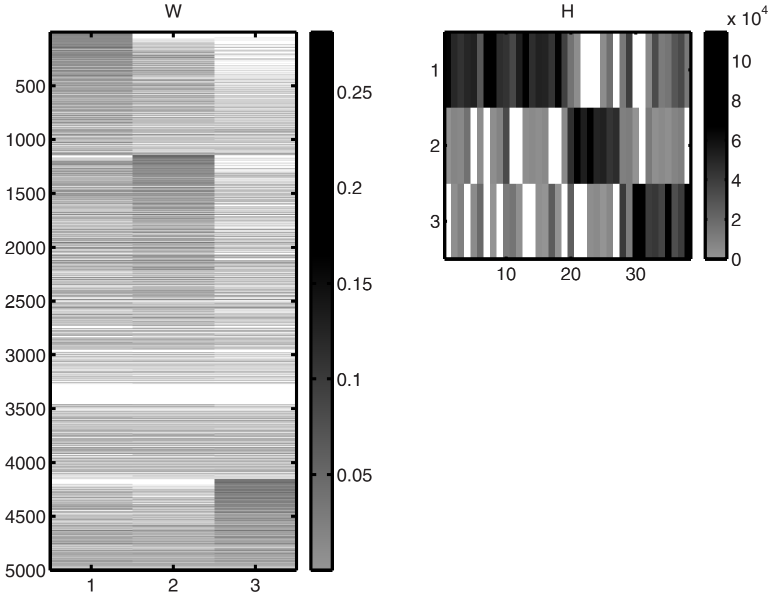


Fig. 4. W (basis matrix) and H (coefficient matrix) obtained from NMF/ANLS for the leukemia dataset (38 samples: 19 ALL-B, 8 ALL-T, 11 AML) with the 5,000 most highly varying genes

However, in most of practical situations, sample j is usually a linear combination of more than one basis vectors. For instance, gene expression pattern under an experimental condition comes from usually more than one biological pathways or processes. K-means gives a centroid-based hard-clustering, while NMF gives a soft-clustering (except the case that each basis vector is a centroid vector).

4 Summary

Cancer class discovery is conducted by using NMF based on alternating non-negativity-constrained least squares, which has a sound convergence property. A rigorous convergence criterion is also introduced. We have established a reliable framework of data analysis using NMF, which is theoretically sound and practically efficient. This approach can be applied to many practical problems in wide areas including text data mining, bioinformatics, and computational biology.

Acknowledgments. This material is based upon work supported in part by the National Science Foundation Grants ACI-0305543 and CCF-0621889. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

1. Lee, D.D., Seung, H.S.: Learning the parts of objects by non-negative matrix factorization. Nature **401** (1999) 788–791
2. Pauca, V.P., Shahnaz, F., Berry, M.W., Plemmons, R.J.: Text mining using non-negative matrix factorizations. In: Proc. SIAM Int'l Conf. Data Mining (SDM'04). (April 2004)
3. Kim, P.M., Tidor, B.: Subsystem identification through dimensionality reduction of large-scale gene expression data. Genome Research **13** (2003) 1706–1718
4. Brunet, J.P., Tamayo, P., Golub, T.R., Mesirov, J.P.: Metagenes and molecular pattern discovery using matrix factorization. Proc. Natl Acad. Sci. USA **101**(12) (2004) 4164–4169
5. Gao, Y., Church, G.: Improving molecular cancer class discovery through sparse non-negative matrix factorization. Bioinformatics **21**(21) (2005) 3970–3975
6. Kim, H., Park, H.: Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares. In Du, D.Z., ed.: Proceedings of the IASTED International Conference on Computational and Systems Biology (CASB2006). (November 2006) 95–100
7. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Proceedings of Neural Information Processing Systems. (2000) 556–562
8. Paatero, P., Tapper, U.: Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics **5** (1994) 111–126
9. Lin, C.J.: Projected gradient methods for non-negative matrix factorization. Technical Report Information and Support Service ISSTECH-95-013, Department of Computer Science, National Taiwan University (2005)
10. Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization (2006) *Computational Statistics and Data Analysis*, to appear.
11. Bro, R., de Jong, S.: A fast non-negativity-constrained least squares algorithm. J. Chemometrics **11** (1997) 393–401
12. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems. Prentice-Hall, Englewood Cliffs, NJ (1974)
13. van Benthem, M.H., Keenan, M.R.: Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. J. Chemometrics **18** (2004) 441–450
14. Gonzales, E.F., Zhang, Y.: Accelerating the Lee-Seung algorithm for non-negative matrix factorization. Technical report, Department of Computational and Applied Mathematics, Rice University (2005)
15. MATLAB: User's Guide. The MathWorks, Inc., Natick, MA 01760 (1992)
16. Golub, T.R., Slonim, D.K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J.P., Coller, H., Loh, M.L., Downing, J.R., Caligiuri, M.A., Bloomfield, C.D., Lander, E.S.: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring. Science **286** (1999) 531–537
17. Ding, C., He, X., Simon, H.D.: On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proc. SIAM Int'l Conf. Data Mining (SDM'05). (April 2005) 606–610

A Support Vector Machine Ensemble for Cancer Classification Using Gene Expression Data

Chen Liao and Shutao Li

College of Electrical and Information Engineering, Hunan University,
Changsha 410082, China
shutao_li@hnu.cn

Abstract. In this paper, we propose a support vector machine (SVM) ensemble classification method. Firstly, dataset is preprocessed by Wilcoxon rank sum test to filter irrelevant genes. Then one SVM is trained using the training set, and is tested by the training set itself to get prediction results. Those samples with error prediction result or low confidence are selected to train the second SVM, and also the second SVM is tested again. Similarly, the third SVM is obtained using those samples, which cannot be correctly classified using the second SVM with large confidence. The three SVMs form SVM ensemble classifier. Finally, the testing set is fed into the ensemble classifier. The final test prediction results can be got by majority voting. Experiments are performed on two standard benchmark datasets: Breast Cancer, ALL/AML Leukemia. Experimental results demonstrate that the proposed method can reach the state-of-the-art performance on classification.

Keywords: Support Vector Machine, Wilcoxon Rank Sum Test, Gene Selection, Ensemble Classifier, Classification Accuracy.

1 Introduction

Accurate cancer diagnosis is crucial for the successful application of specific therapies nowadays. Recent studies show that DNA microarrays can provide useful information for cancer classification at the gene expression level because of their ability to measure the large quantity of messenger ribonucleic acid (mRNA) transcripts for genes simultaneously [1].

There are many classification methods used in cancer diagnosis. Such as K-nearest neighbor (KNN), neural network, SVM and so on. KNN uses an integer parameter *K*. Provided an input, the algorithm finds the *K* closest training data points to the input, and then the label of the input based on the label of the *K* points will be predicted [2]. Neural network is also often used in cancer classification. It contains some structure consisting of a certain number of hidden layers and one output layer. Among the different training procedures of neural networks, the back propagation algorithm with adaptive learning and momentum is most popular and is often utilized. The learning algorithm will not be stopped until the classification performance of the validation set starts to diverge from that of the training set [3]. SVM is one of the most popular

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 488–495, 2007.

cancer classification methods. If the data is linearly separable, it computes the hyperplane which maximizes the margin between the training samples and the class boundary. And when the data is not linearly separable, the samples are projected into a high dimensional space where such a separating hyperplane can be found [4].

SVM ensemble is actually a type of cross-validation optimization of single SVM with a more stable classification performance than other models. In this paper, a SVM ensemble classification method is proposed. We construct SVM ensemble classifier. The first SVM uses all of the training samples, and the next two SVMs respectively use the samples, which are incorrectly classified or low confidence obtained by the preceding SVM.

This paper is organized as follows. In next section, the basic theory of SVM is introduced. In section 3, the new classification method is proposed. In section 4, the experimental results are shown, and in the last section, the paper is concluded.

2 Support Vector Machines

SVM has been considered as a widely used classification approach of statistical learning theory. The training set is supposed to be$\{(x_i, y_i)\}_{i=1}^N$, with each input $x_i \in R^m$ and $y_i \in \{\pm 1\}$. The SVM projects x to $z = \varphi(x)$ in a Hilbert space F by a nonlinear map $\varphi : R^m \to F$. The dimensionality of F is very high in most conditions. When the data is linearly separable in F, a hyperplane $(\langle w, \varphi(x)\rangle + b)$ is constructed by the SVM, the separation between the positive and negative examples is maximized for the hyperplane. By minimizing ||*w*||, the *w* for the optimal hyperplane is obtained, and the solution can be presented as $w = \sum_{i=1}^N \alpha_i y_i \varphi(x_i)$ for some certain $\alpha_i \geq 0$. The vector of α_i's, $\Lambda = (\alpha_i, ..., \alpha_N)$, can be obtained by solving the following quadratic programming problem:

$$\text{maximize } W(\Lambda) = \Lambda^T 1 - \frac{1}{2}\Lambda^T Q \Lambda \text{ ,} \tag{1}$$

with respect to Λ, subject to the constraints $\Lambda \geq 0$ and $\Lambda Y = 0$. Here, $Y^T = (y_1, ..., y_N)$ and Q are symmetric matrixes with elements

$$Q_{ij} = y_i y_j \langle \varphi(x_i), \varphi(x_j) \rangle \text{ .} \tag{2}$$

For those α_i's greater than zero, the relevant training examples should lie along the margins of the decision boundary, and these are defined as the support vectors.

However, due to the high dimensionality of F and $\varphi(x_i)$and $\varphi(x_j)$ in (2), the way is not so impractical. An important characteristic of the SVM, and of kernel methods in general, plays a crucial part here. It is that one can gain $\langle \varphi(x_i), \varphi(x_j) \rangle$ in (2) without explicit $\varphi(x_i)$ and $\varphi(x_j)$ first, this is realized by using kernel function. The kernel methods provide good tools to process, analyze, and compare many types

of data, and provide state-of-the-art performance in many cases. Here some kinds of kernel functions are introduced as follows:
① linear kernel

$$K_L(x, y) = x^T y \ , \tag{3}$$

where x is the value of the independent variable for which one seeks an estimate, and y are the values of the independent variable in the data.
② polynomial kernels

$$K_P(x, y) = (x^T y)^d \ , \tag{4}$$

where d is the degree of the polynomial, its kernel K_p of degree 2 is corresponding to a feature space spanned by all products of two variables, that is, $\{x_1^2, x_1 x_2, x_2^2\}$.
③ Gaussian RBF kernel

$$K_G(x, y) = \exp(-\sigma * \|x - y\|^2) \ , \tag{5}$$

where σ is a parameter, the Gaussian kernel is one of the most popular utilized kernels in practice due to its capacity to produce nonparametric classification functions [5].

3 Proposed Method

We suppose a gene expression dataset with M genes (features) and N samples (observations) be represented by the following matrix:

$$\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1N} \\ x_{21} & x_{22} & \cdots & x_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ x_{M1} & x_{M2} & \cdots & x_{MN} \end{bmatrix} ,$$

where x_{ij} is the measurement of the expression level of gene i in sample j. Let $\mathbf{x}_j = (x_{1j}, x_{2\mathrm{j}}, ..., x_{Mj})$ denote the jth sample of X, and y_j the corresponding class label (e.g, tumor type or clinical outcome).

The schematic diagram of the proposed method is shown in Figure 1.

Step 1: Preprocessing using Wilcoxon rank sum test
Due to the small size of training set, leave-one-out cross validation (LOOCV) is utilized to select the training set and the testing set.

The statistics formula of Wilcoxon rank sum test is:

$$s(g) = \sum_{i \in \mathbf{N}_{+1}} \sum_{j \in \mathbf{N}_{-1}} I((\mathbf{x}_j^{(g)} - \mathbf{x}_i^{(g)}) \leq 0) \ ,$$

where I is the discrimination function, if the logic expression in the bracket is true, the value of I is 1, or else it is 0. $\mathbf{x}_i^{(g)}$ is the expression value of the sample i in the gene g. $\mathbf{N}_{+1}$ and $\mathbf{N}_{-1}$ are the index sets of different classes of samples. $s(g)$ can represent the measurement of the difference between the two classes. When it is closer to 0 or closer to the max value of $n_{+1}n_{-1}$ (here $n_{+1} = |\mathbf{N}_{+1}|$, $n_{-1} = |\mathbf{N}_{-1}|$), the corresponding gene is more important to the classification. So according to (6), the importance degree of gene can be calculated [6]:

$$q(g) = \max(s(g), n_{-1}n_{+1} - s(g)) \ . \tag{6}$$

Genes are ranked according to $q(g)$, and the top m genes are selected to form a new training subset.

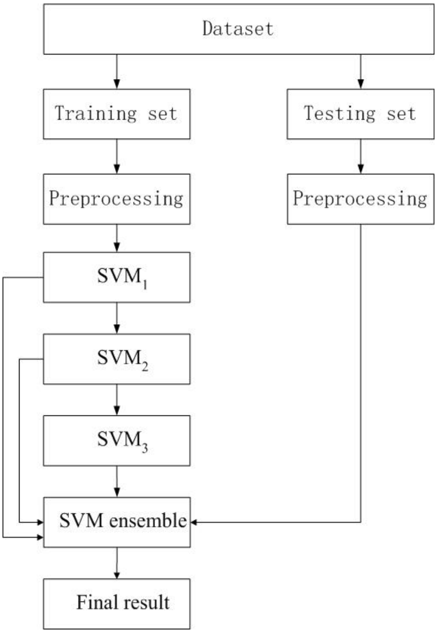


Fig. 1. The proposed method of SVM ensemble classification

Step 2: Constructing SVM ensemble classifier

Using each new training set, we train SVM$_1$, and SVM$_1$ is tested by the training set itself to obtain the prediction results. Then we select the samples with incorrect prediction result or low confidence to train SVM$_2$, and SVM$_2$ is also tested. As the same, SVM$_3$ can be obtained. Finally, SVM ensemble classifier is composed of these three SVM together.

The value of confidence is defined as:

$$f(x) = \left| w^T \varphi(x) + b \right| \cdot$$

And the dual form is

$$f(x) = \left| \sum_{j \in SV} \alpha_j y_j K(x_j, x) + b \right| ,$$

where α_j is Lagrange multiplier, and y_j represents for the *j*th support vector.

Step 3: Classification

The testing set is fed into the ensemble classifier, and there are three classification results from the three SVM classifiers. We use majority voting to decide the final classification result, that's to say, if more than one classifier show the result to be +1, the final classification result will be +1, otherwise it will be -1.

4 Experimental Setup and Results

4.1 Setup

The gene selection and classification performance of the proposed method is evaluated by two benchmark datasets: Breast cancer dataset: It consists of a total of 38 samples, 18 of them are ER+ (estrogen receptor) samples while the remaining 20 are ER-. Each sample contains expression values of 7129 genes [7]. ALL/AML Leukemia dataset: It contains a total of 72 samples of two types of leukemia: 25 of acute myeloid leukemia (AML) and 47 of acute lymphoblastic leukemia (ALL). Each sample contains expression values of 7129 genes [8].

4.2 Experimental Results

In our experiments, the samples with error classification result or confidence less than 1 are used to construct the next classifier. For Breast cancer dataset, the numbers of training sample of SVM_1, SVM_2 and SVM_3 are respectively 37, 13 and 11. And for Leukemia dataset, the numbers are respectively 71, 17 and 16.

Table 1 shows the classification accuracy of Breast cancer dataset with different parameters, and Table 2 shows the results of Leukemia dataset.

In these tables, *m* is the number of the informative genes selected by Wilcoxon rank sum test. Gaussian kernel is used and there are two parameters σ and C for each SVM. σ_1, C_1, σ_2, C_2, σ_3 and C_3 represent the corresponding σ and C for SVM_1, SVM_2 and SVM_3 , respectively.

Because σ_1 and C_1 have the most obvious effect on the classification results, we only list the effect of σ_1 and C_1 in the tables for the length limit.

Table 3 shows the classification accuracy obtained by single SVM classifier, and Gaussian kernel is also used with the value of σ set to 0.0001. For Breast cancer, it can reach the prediction result of 100%, and for Leukemia dataset, it's only 97.2%. Compared to the results using single SVM classifier, we can conclude that SVM ensemble classifier can obtain better and more stable prediction results.

Table 4 shows the performance of various methods on the two datasets as reported in the literature. All these methods use LOOCV and so their classification accuracies can be directly compared. As can be seen, the proposed method, which attains the best classification accuracy (of 100%) on both of Breast cancer dataset and Leukemia dataset, outperforms most of the methods. The JCFO (Joint Classifier and Feature Optimization) [9] with linear kernel can also attain 100% on Leukemia dataset, however, JCFO relies on the Expectation-Maximization (EM) algorithm [9] and is much slower.

As can be seen, for both Breast cancer and Leukemia data, SVM ensemble can reach the best classification performance 100%.

Table 1. Classification accuracy (%) of Breast cancer data (σ_2 =0.1, C_2=10, σ_3=0.01, C_3=100)

m	σ_1	C_1	Classification accuracy
50	0.1	10	**100.0**
		100	**100.0**
	0.01	10	**100.0**
		100	**100.0**
	0.001	10	**100.0**
		100	**100.0**
70	0.1	10	**100.0**
		100	**100.0**
	0.01	10	**100.0**
		100	**100.0**
	0.001	10	**100.0**
		100	**100.0**
90	0.1	10	**100.0**
		100	**100.0**
	0.01	10	**100.0**
		100	**100.0**
	0.001	10	**100.0**
		100	**100.0**

Table 2. Classification accuracy (%) of Leukemia data (σ_2 =0.001, C_2=10, σ_3=0.01, C_3=10)

m	σ_1	C_1	Classification accuracy
50	0.01	1	98.6
		10	98.6
	0.001	1	98.6
		10	98.6
	0.0001	1	98.6
		10	98.6
70	0.01	1	97.2
		10	97.2
	0.001	1	97.2
		10	97.2
	0.0001	1	97.2
		10	97.2
90	0.01	1	**100.0**
		10	98.6
	0.001	1	**100.0**
		10	98.6
	0.0001	1	**100.0**
		10	98.6

Table 3. Classification accuracy (%) obtained by single SVM classifier

m	*C*	Breast cancer	Leukemia
30	1	52.6	65.3
	100	89.5	81.9
	1000	**100.0**	94.4
50	1	52.6	65.3
	100	**100.0**	88.9
	1000	**100.0**	97.2
70	1	52.6	65.3
	100	**100.0**	93.1
	1000	**100.0**	97.2
90	1	52.6	65.3
	100	**100.0**	95.8
	1000	**100.0**	97.2

Table 4. Classification accuracy (%) obtained by various methods

Classifier	Breast cancer	Leukemia
Adoboost (decision stumps) [10]	--	95.8
SVM (quadratic kernel) [10]	--	95.8
SVM (linear kernel) [10]	97.4	94.4
RVM (linear kernel) [9]	94.7	94.4
RVM (no kernel) [9]	89.5	97.2
Logistic regression (no kernel) [9]	--	97.2
Sparse probit regression (quadratic kernel) [9]	--	95.8
Sparse probit regression (linear kernel) [9]	97.4	97.2
Sparse probit regression(no kernel) [9]	84.2	97.2
JCFO (quadratic kernel) [9]	--	98.6
JCFO (linear kernel) [9]	97.4	**100.0**
Proposed method	**100.0**	**100.0**

5 Conclusion

A SVM ensemble classification method is proposed in this paper. Experiments are performed on the Breast cancer and Leukemia datasets. While the use of single SVM classifier does not yield satisfactory results, the ensemble classifier shows the superior classification performance on both of the datasets, and the classification performance of both datasets can reach to a state-of-the-art level. The SVM ensemble classification method proves to be a reliable method.

Acknowledgement

This paper is supported by the National Nature Science Foundation of China (No. 6040204), Program for New Century Excellent Talents in University, and the Excellent Youth Foundation of Hunan Province (06JJ1010).

References

1. Wang, Y., Tetko, I. V., Hall, M. A., Frank. E., Facius. A., Mayer, K. F. X., Mewes, H. W.: Gene Selection from Microarray Data for Cancer Classification—A Machine Learning Approach. Computational Biology and Chemistry 29 (2005) 37-46
2. Li, T., Zhang, C., Ogihara, M.: A Comparative Study of Feature Selection and Multiclass Classification Methods for Tissue Classification Based on Gene Expression. Bioinformatics 20 (2004) 2429-2437
3. Guo, H., Jack, L. B., Nandi, A. K.: Feature Generation Using Genetic Programming with Application to Fault Classification. IEEE Transactions on Systems 35 (2005) 89-99
4. Huerta, E. B, Duval, B., Hao J.: A Hybrid GA/SVM Approach for Gene Selection and Classification of Microarray Data. EvoWorkshops, Budapest (2006) 34-44
5. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines, Cambridge University Press, Cambridge (2000)
6. Park, P. J., Pagano, M., Bonetti, M.: A Nonparametric Scoring Algorithm for Identifying Informative Genes from Microarray Data. Pacific Symposium on Biocomputing (2001) 52-63
7. West, M., Blanchette, C., Dressman, H., Huang, E., Ishida, S., Spang, R., Zuzan, H., Marks, J. R., Nevins, J. R.: Predicting the Clinical Status of Human Breast Cancer Using Gene Expression Profiles. Proceedings of the National Academy of Science 98 (2001) 11462-11467
8. Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. H. H. C., Loh, M., Downing, J., Claligiuri, M., Bloomfield, C., Lander, E.: Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. Science 286, 531-537
9. Krishnapuram, B., Carin, L., Hartemink, A.: Gene Expression Analysis: Joint Feature Selection And Classifier Design. In Kernel Methods in Computational Biology, Schölkopf, B., Tsuda, K., & Vert, J, -P., eds. MIT Press (2004)
10. Ben-Dor, A., Bruhn, L., Friedman, N., Nachman, I., Schummer, M., Yakhini, Z.: Tissue Classification with Gene Expression Profiles. Proceedings of the Fourth Annual International Conference on Computational Molecular Biology (2000) 54-64

Combining SVM Classifiers Using Genetic Fuzzy Systems Based on AUC for Gene Expression Data Analysis

Xiujuan Chen[1], Yichuan Zhao[2], Yan-Qing Zhang[1], and Robert Harrison[1]

[1] Department of Computer Science, Georgia State University, Atlanta, GA 30302, USA
[2] Department of Mathematics and Statistics, Georgia State University, Atlanta, GA 30302, USA
{xchen8@,matyiz@langate,zhang@taichi.cs, cscrwh@asterix.cs}gsu.edu

Abstract. Recently, the use of Receiver Operating Characteristic (ROC) Curve and the area under the ROC Curve (AUC) has been receiving much attention as a measure of the performance of machine learning algorithms. In this paper, we propose a SVM classifier fusion model using genetic fuzzy system. Genetic algorithms are applied to tune the optimal fuzzy membership functions. The performance of SVM classifiers are evaluated by their AUCs. Our experiments show that AUC-based genetic fuzzy SVM fusion model produces not only better AUC but also better accuracy than individual SVM classifiers.

Keywords: Receiver Operating Characteristic (ROC), Support Vector Machines (SVMs), Gene Expression, Genetic Fuzzy System (GFS), Classifier fusion.

1 Introduction

With the key technologies developed in the biomedical area, such as DNA sequencing, micro-array, and structure genomics, large-scale biological and biomedical data have been accumulated, including DNA sequences, protein sequences and structures, gene expression data, protein profiling data, and genomic sequence data. Accordingly, the bulk of research efforts have been shifted to the biomedical data analysis to extract patterns and discover useful information from the data and therefore provide valuable supports for biomedical and evolutionary research.

Machine learning and classification techniques have been widely used to assist the interpretation and analysis of biomedical data. As recently discovered pattern recognition tools, Support Vector Machines (SVMs) [1] have become popular become of their outstanding learning performance when applied to real-world classification applications. SVMs are capable of classifying not only linear separable but also non-linear separable problems. They aim to find an optimal hyperplane to separate positive/negative classes with the maximum margin in a high dimensional feature space, which is transformed from the original input space by applying a kernel function, for instance, a polynomial or a RBF kernel.

However, how to select an appropriate SVM kernel to achieve the best possible performance for a real classification application is one of the practical difficulties.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 496–505, 2007.

Instead of finding the best kernel for the application by exhaustively trying out all possible kernel functions with all possible parameters, classifier fusion methods provide a more efficient but still high-performance way of solving the practical problem existing in the SVM classification.

Classifier fusion is to combine a set of classifiers in a certain way so that the combined classifier can receive a better performance than its composing individual classifiers. The reason that the combined classifier could outperform the best individual classifier is because the data examples misclassified by the different classifiers would not necessarily overlap, which leaves the room for the classifier complementariness [2]. One sufficient condition for a combined classifier to be more accurate than any of its individual members is that individual classifiers should be *accurate* and *diverse* [3]. "Diverse" is crucial when combining classifiers. Different classifiers can be achieved by using different data feature sets, different training sets, or different classification algorithms [2], [3], [4], [5], [6].

In this study, we propose a classifier fusion model particularly for SVM classifiers aiming to boost the performance of SVM classifiers. A fuzzy logic system (FLS) is constructed to combine multiple SVM classifiers in the light of the performance of each individual classifier. The memberships of the fuzzy logic system are tuned by genetic algorithms (GAs) to generate the optimal fuzzy logic system.

One question here is how to evaluate classifier performance in the fusion model. Typically, *accuracy* is the standard criterion to evaluate a classifier performance [7], [8]. In many scenarios, however, accuracy is not enough or not much meaningful. Researchers are often interested in ranking of data examples rather than mere positive/negative classification results. Moreover, if class distribution is skewed or unbalanced, a classifier can still receive a high accuracy by simply classifying all data examples in the dominant class [7], [9]. Recently, the Receiver Operating Characteristics (ROC) and the area under an ROC curve (AUC) have been shown to be statistically consistent with and more discriminating than accuracy empirically and theoretically [7], [8], [10]. This paper will use AUC as the evaluation of classifier performance to build the genetic fuzzy fusion model to enhance the performance of SVM classifiers. It has also been shown that classifiers based on AUC produce not only better AUC, but also better accuracy [11].

In this paper, we will first introduce the concepts associated with ROC analysis in Section 2. Then we will discuss genetic fuzzy algorithms in Section 3. The SVM classifier fusion model will be proposed in Section 4 and gene expression data will be experimented in Section 5. Finally in Section 6, conclusions will be drawn.

2 ROC Analysis for Binary Classification

ROC has been receiving much attention recently as a measure to analyze classifier performance and has attractive properties that make it especially useful for domains with skewed class distribution and unequal classification error costs.

An ROC curve of a classifier is a plot of true positive rate (*TPR*) on *Y* axis versus false positive rate (*FPR*) on the *X* axis as shown in Fig.1. The *TPR* and *FPR* are defined as follows [9].

$$TPR = \frac{TP}{N^{+}}, \qquad FPR = \frac{FP}{N^{-}} \tag{1}$$

where *TP* denotes true positives, *FP* denotes false positives, and N^+ and N^- denote positives and negatives respectively.

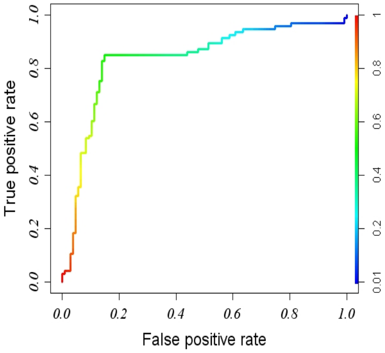


Fig. 1. An ROC curve

For a *discrete classifier*, which produces only a positive/negative class label on each example, only a single point can be drawn in the ROC graph. However, for a *probabilistic classifier*, which yields a numeric value on each example representing the degree to which an example belongs to a class, if various decision thresholds are applied to classify data examples, a series of points can be plotted in a ROC plane with pairs of {*FPR*, *TPR*} as their coordinates. Each threshold results in one point on the ROC curve representing the classifier which is generated by using this threshold as the cutoff point. Therefore, an ROC curve of a probabilistic classifier can be viewed as an aggregation of classifiers from all possible decision thresholds [7].

The quality of an ROC curve can be summarized in one value by calculating the area under the ROC curve (AUC). AUC represents the probability that one classifier ranks a randomly chosen positive example higher than a randomly chosen negative example [9]. According to Hand [12], AUC can be simply calculated in the following formula:

$$AUC = \frac{\sum_{i=1}^{N^+} r_i - N^+(N^+ + 1)/2}{N^+ N^-} \tag{2}$$

where r_i denotes the rank of *i*th positive example in the ranking list if we arrange the classification results of data examples in ascending order.

AUC has been shown to be a better measure than accuracy when assessing classifier performances [8], [10].

3 Genetic Fuzzy Systems

Fuzzy logic has demonstrated the powerful abilities to handle the imprecision and uncertainties in real-world applications. It captures uncertainties by defining linguistic fuzzy sets with fuzzy membership functions (MFs) and reasoning fuzzy rules in a rigorous mathematical discipline. However, the success of designing a fuzzy logic system (FLS) largely relies on high-performance fuzzy MFs and fuzzy rules to interpret the expert knowledge. When lack of human expert, rather than choosing fuzzy MFs or defining fuzzy rules in a manual trail-and-error manner, we may seek the assistant from a learning process.

A genetic fuzzy system (GFS) is able to learn and search fuzzy MFs or fuzzy rules efficiently. It is basically a fuzzy system augmented by a learning process based on a genetic algorithm [13]. GAs are optimization algorithms inspired by natural evolution and provide robust search and learning capabilities in complex space. GAs are able to learn or train or tune different components of fuzzy logic systems [13]. For instance, some genetic fuzzy rule-based systems may learn and determine the number of IF-THEN fuzzy rules from all possible rules [14], [15]. Other genetic fuzzy systems may tune MFs of a given fuzzy rule set, such as tuning positions or shapes of MFs [16], [17], [18].

To tune fuzzy MFs, there are two techniques in general: Pittsburgh approach and Michigan approach [13]. Pittsburgh approach is to represent an entire fuzzy rule set as a chromosome and maintain a population of candidate rule sets using genetic operations to produce new generations of rule sets [19]. Michigan approach is to represent an individual rule as a chromosome and the whole rule set is represented by the entire population [20].

4 Genetic Fuzzy SVM Fusion Based on AUC

The genetic fuzzy fusion model for combing SVM classifiers is constructed as shown in Fig. 2. The system has three phases. In phase I, training data are trained on

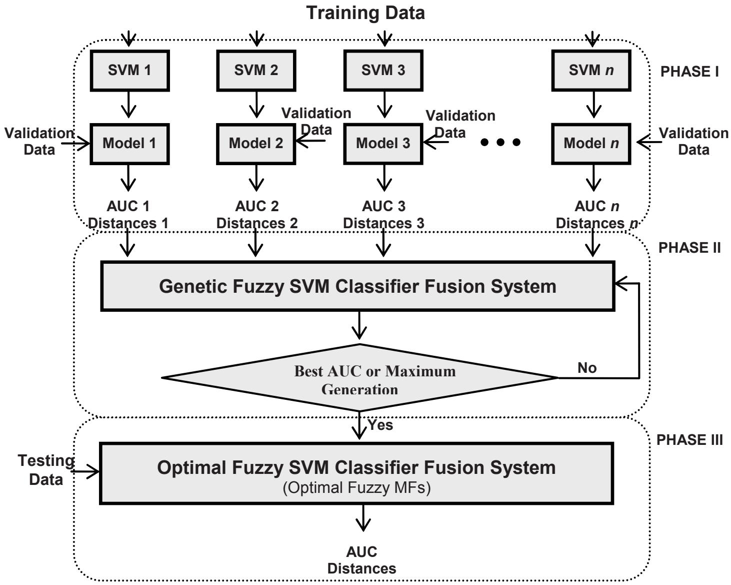


Fig. 2. Genetic fuzzy fusion system for SVM classifiers

different SVMs. Validation data are classified to obtain individual SVM AUCs and distances of validation data examples to SVM hyperplanes. In phase II, a GFS is constructed and fuzzy MFs are tuned by GAs in cross validation manner. Finally, in phase III, testing data are fed into the optimal fuzzy fusion system to make the final decision. We have implemented the proposed fusion system on combing THREE SVM classifiers and will give the detailed explanation in the rest of the section. This process can be easily extended to combine arbitrary number of SVMs in general.

4.1 Fuzzy System Inputs and Output

The fuzzy fusion system is designed by applying Mamdani model [21] where the consequences of fuzzy rules are fuzzy sets. In the fusion system combining three SVM classifiers, there are three *AUC* inputs depicting three SVM classifier performances, three *distance* inputs representing the classification results of a data example from three individual SVM classifiers, and one output indicating the final decision from the fusion system for the example.

All the MFs of the inputs and output are defined as simple triangles shown in Fig. 3. Each AUC input is described by two fuzzy sets: low and high, and each

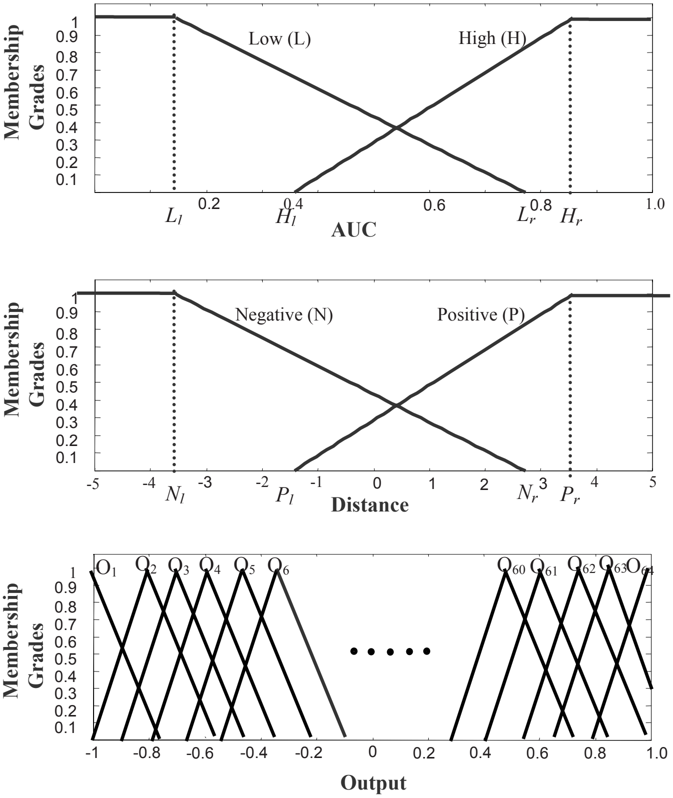


Fig. 3. MFs for the inputs and output

distance input is also represented by two fuzzy sets: negative and positive. The output is composed of 64 fuzzy sets corresponding to the consequences of 64 fuzzy rules. All the MFs are not fixed and each has control parameters to control its position and shape. Each AUC MF or distance MF has two control points and each output MF has one control point. We will discuss how to tune the MFs later.

4.2 Fuzzy Rule Base

There are 64 rules in total each corresponding to one of 64 combinations of six inputs (2 ^ 6 = 64). The *i*th (*i* = 1...64) fuzzy rule is defined as follows:

> **IF** ***auc*$_1$** is **A$_{i1}$** and ***auc*$_2$** is **A$_{i2}$** and ***auc*$_3$** is **A$_{i3}$** and ***dis*$_1$** is **D$_{i1}$** and ***dis*$_2$** is **D$_{i2}$** and ***dis*$_3$** is **D$_{i3}$**, **THEN *g$_i$*** is **O$_i$** (*i* = 1...64).

where auc_j denotes *j*th AUC input and dis_j denotes *j*th distance input (*j*=1..3). A_{ij} (*j*=1..3) denotes the AUC fuzzy set in {Low, High}, D_{ij} (*j*=1..3) denotes a distance fuzzy set in {Negative, Positive}, and O_i denotes an output fuzzy set in {O_1...O_{64}} for the *i*th rule.

4.3 Fuzzy System Output and Defuzzification

The system output is calculated by aggregating individual rule contributions:

$$y = \sum_{i=1}^{64} \beta_i g_i \Big/ \sum_{i=1}^{64} \beta_i \tag{3}$$

where g_i is the output value of the *i*th rule and β_i is the firing strength of the *i*th rule defined by product *t*-norm:

$$\beta_i = \prod_{j=1}^{3} \mu_{A_{ij}}(auc_j) * \mu_{D_{ij}}(dis_j) \tag{4}$$

where $\mu_{A_{ij}}(auc_j)$ and $\mu_{D_{ij}}(dis_j)$ are the membership grades of input auc_j and dis_j (*j*=1…3) in the fuzzy sets A_{ij} and D_{ij}.

If the output value is greater than or equal to 0, the data example is defuzzified in the positive class. Otherwise, it is in the negative class. This information may be used to calculate the accuracy of the model.

4.4 Tuning Fuzzy System Using GAs

We use a real-coded GA and apply Pittsburgh approach [19] to tune the input and output MFs. Each chromosome is composed of the 72 genes representing 72 entire membership control parameters: 4 control points for AUC, 4 for distance MFs, and the rest 64 for output MFs.

The fuzzy MFs are tuned in cross-validation manner. The fitness of the GA is defined to maximize the average AUC of each fold of data examples by applying the same MFs defined in a chromosome.

Selection schema is roulette-wheel selection, which implies the higher the AUC, the greater the chance that a chromosome will be selected into the next generation. We also apply elitism strategy to ensure the best fuzzy MFs to be selected. Uniform crossover is used here since it is believed to outperform one or multiple crossover in many applications. We apply Gaussian mutation to modify genes by adding a Gaussian distributed random number with a mean of zero to them [22].

5 Experiments on Gene Expression Data

In the experiment, we have tested the proposed SVM fusion model using colon tumor dataset from Kent Ridge Biomedical Data Set Repository [23]. The colon tumor dataset is a set of gene expression data. It is collected from colon cancer patients. There are totally 62 data examples, among which 40 examples are tumor tissues from diseased parts of the patients and 22 are normal tissues from healthy parts of the colons of the same patients. Each example is composed of 2000 genes as 2000 features.

The data in Phase I in Fig. 2 are classified using SVM^{Light} software [24]. The generalization parameter *C* of SVMs is set to 1. Two types of kernels are used: polynomial kernels and RBF kernels. The degree in polynomial kernels are set to 1, 2, …, 10, and gamma in RBF kernels are set to 10^{-4}, 10^{-3},…,10^{1}. In order to avoid the selection bias, we apply cross validation strategy to assess accuracies and AUCs of individual SVM classifiers. Table 1 shows the SVM testing AUCs and testing accuracies for colon tumor data in 4-fold cross validation.

The genetic fuzzy system is constructed and tuned in cross-validation manner as well. Each training dataset in Phase I is further divided into second-level training and testing data. The second-level training data are still trained by SVM^{Light} to obtain the AUCs and distances of the second-level testing data (validation data in Fig. 2), which will be used as the inputs of the genetic fuzzy fusion system to tune the optimal fuzzy MFs based on AUC measure. After the optimal MFs are adapted, the testing data in the first-level are applied to the tuned optimal fuzzy fusion model to make the final decision. The genetic fuzzy fusion system combining three SVM classifiers has been implemented in C language. The parameter setting for the GA is as follows: crossover probability 90%, generation of 200, and population size of 3000.

Table 1. Testing AUC and accuracy using individual SVMs (4-fold cross-validation)

Kernels		Testing AUC					Testing Accuracy (%)				
Polynomial	degree	1	2	3	4	Avg.	1	2	3	4	Avg.
poly_1	1	0.87	0.94	0.84	0.91	0.89	75.00	93.75	86.67	80.00	83.86
poly_2	2	0.87	0.90	0.86	0.91	0.88	81.25	87.50	86.67	86.67	85.52
poly_3	3	0.87	0.84	0.82	0.77	0.82	75.00	75.00	86.67	80.00	79.17
poly_4	4	0.88	0.83	0.76	0.77	0.81	87.50	81.25	66.67	66.67	75.52
poly_5	5	0.88	0.84	0.70	0.73	0.79	87.50	81.25	66.67	66.67	75.52
poly_6	6	0.90	0.83	0.70	0.70	0.78	87.50	75.00	66.67	66.67	73.96
poly_8	8	0.85	0.79	0.64	0.68	0.74	87.50	75.00	66.67	66.67	73.96
poly_10	10	0.85	0.73	0.56	0.68	0.71	87.50	75.00	66.67	66.67	73.96
RBF	gamma										
rbf_0.0001	0.0001	0.83	0.95	0.84	0.91	0.88	62.50	56.25	66.67	73.33	64.69
rbf_0.001	0.001	0.85	0.95	0.82	0.91	0.88	62.50	56.25	66.67	73.33	64.69
rbf_0.01	0.01	0.85	0.95	0.82	0.91	0.88	62.50	56.25	66.67	73.33	64.69
rbf_0.1	0.1	0.87	0.94	0.82	0.91	0.88	62.50	56.25	66.67	73.33	64.69
rbf_1	1	0.83	0.90	0.78	0.93	0.86	75.00	93.75	86.67	73.33	82.19
rbf_10	10	0.85	0.90	0.74	0.82	0.83	62.50	56.25	66.67	80.00	66.36

We chose six groups of three SVM classifiers and combined each group of three SVMs using the proposed genetic fuzzy SVM fusion model. Table 2 shows the experimental results.

Table 2. Three selected SVM classifiers, maximum and average AUC and accuracy (%) of the three individual SVM classifiers, and AUC and accuracy (%) from the fusion model by combing the three SVMs

Test	SVM1	SVM2	SVM3	Max AUC	Avg. AUC	Max Accuracy	Avg. Accuracy	Fusion AUC	Fusion Accuracy
1	poly_1	poly_3	rbf_0.01	0.89	0.86	83.86	75.91	0.92	88.75
2	poly_4	poly_8	rbf_1	0.86	0.80	82.19	77.22	0.87	83.75
3	poly_4	rbf_1	rbf_0.01	0.88	0.85	82.19	74.13	0.91	85.11
4	poly_1	poly_5	poly_10	0.89	0.80	83.86	77.78	0.89	86.94
5	rbf_0.0001	rbf_0.01	rbf_1	0.88	0.87	82.19	70.52	0.88	88.81
6	poly_3	rbf_0.001	rbf_0.1	0.88	0.86	79.17	69.52	0.89	88.65
Avg.				**0.880**	**0.841**	**82.243**	**74.181**	**0.893**	**87.002**

From Table 2 we may see that the proposed SVM fusion model demonstrates stable and robust classification capabilities. It not only performs far better than the average of three individual SVM classifiers in terms of both AUC and accuracy, but also outperforms the best of three individual SVMs in terms of accuracy and achieves as least as much performance as the best in terms of AUC. For all the six tests, the model accuracy is better than the best accuracy. For Tests 1, 5 and 6, the model achieves 88% accuracy, but the best accuracy is only no more than 83%. For four of six tests (Tests 1, 2, 3, 6), the model achieves a better AUC than the best AUC of three individual SVM classifiers. The remaining (Tests 4, 5) receives the same AUC as the best. These two tests combine three RBF or three polynomial SVM classifiers. RBF classifiers or polynomial classifiers behavior similar and this might cause not much complementary room for the combined classifiers.

We can also see that the classifier fusion model that optimizes AUC measure not only achieves nice AUC performance, but also excellent accuracy as well [11]. The genetic fuzzy SVM fusion model based on AUC produces a combined classifier with the best AUC naturally because of the properties of AUC. This means that the accurate ranking of data examples is maintained and it provides researchers more interpretation of the data than mere positive or negative classification results.

6 Conclusion

In this paper, we propose a genetic fuzzy SVM classifier fusion model to combine multiple SVM classifiers. Individual SVMs are combined in a genetic fuzzy system and GAs are applied to tune the fuzzy MFs based on AUC measure. The experimental results show that the proposed genetic fuzzy system is more stable and more robust than individual SVMs. Moreover, the combined SVM classifier from the genetic fuzzy fusion model accomplishes more accurate ranking of data examples which provides valuable interpretation of the real-world data and may help medical diagnosis.

Acknowledgments. This work was supported in part by NIH under P20 GM065762, NIGMS 065762, Georgia Cancer Coalition, and Georgia Research Alliance. Dr. Harrison is a GCC distinguished cancer scholar.

References

1. Vapnik, V. N.: The Nature of Statistical Learning Theory. Springer-Verlag, New York (1995)
2. Kittler, J., Hatef, M., Duin R., Matas J.: On Combining Classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 20, No. 3. (1998) 226-239
3. Hansen, L., Salamon, P.: Neural network ensembles, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 12. (1990) 993-1001
4. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision Combination in Multiple Classifier Systems, IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 16, No. 1. (1994) 66-75
5. Ho, T.K.: Random Decision Forests, Third Int'l Conf. Document Analysis and Recognition, Montreal. (1995) 278-282
6. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition, IEEE Trans. Systems, Man, and Cybernetics, Vol. 22, No. 3. (1992) 418-435
7. Qin, Z.-C.: ROC Analysis for Predictions Made by Probabilistic Classifiers, Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Vol. 5. (2005) 3119-3124
8. Ling, C.X., Huang, J., Zhang, H.: AUC: A Statistically Consistent and More Discriminating Measure than Accuracy, Proc. 18th Int'l Conf. Artificial Intelligence (IJCAI '03). (2003) 329-341
9. Fawcett, T.: ROC graphs: Notes and practical considerations for researchers, Tech Report HPL-2003-4, HP Laboratories. (2003)
10. Huang, J., Ling, C.X.: Using AUC and Accuracy in Evaluating Learning Algorithms. IEEE Trans. Knowl. Data Eng, Vol. 17, No. 3. (2005) 299-310
11. Ling, C.X., Zhang, H.: Toward Bayesian Classifiers with Accurate Probabilities, Proceedings of the Sixth Pacific-Asia Conference on KDD, Springer. (2002)
12. Hand, D. J., Till, R. J.: A Simple Generalization of the Area under the ROC Curve for Multiple Class Classification Problems, Machine Learning, Vol. 45. (2001) 171–186
13. Magdalena, L., Cordon, O., Gomide, F., Herrera, F., Hoffmann, F.: Ten Years of Genetic Fuzzy Systems: Current Framework and New Trends, Fuzzy Sets & Systems, Vol. 141, No. 1. (2004) 5-31.
14. Herrera, F., Lozano, M., Verdegay, J.L.: Generating Fuzzy Rules from Examples Using Genetic Algorithms, Fuzzy Logic and Soft Computing. (1995c)
15. 15. Karr, C.: Applying Genetic to Fuzzy Logic, AI Expert, Vol. 6. (1991) 26-33
16. Homaifar, and McCormick, E.: Simultaneous Design of Membership Functions and Rule Sets for Fuzzy Controllers Using Genetic Algorithms, IEEE Transactions on Fuzzy Systems, Vol. 3, No. 2. (1995) 129-139
17. Park, D., Kandel, A.: Genetic-based New Fuzzy Reasoning Models with Application to Fuzzy Control, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 24, No. 1. (1994) 39-47
18. Cordon and Herrera, F.: A Three-Stage Evolutionary Process for Learning Descriptive and Approximate Fuzzy Logic Controller Knowledge Bases from Examples, International Journal of Approximate Reasoning, Vol. 17, No. 4. (1997) 369-407

19. Smith, S.: A Learning System Based on Genetic Adaptive Algorithms, Doctoral dissertation, Department of Computer Science, University of Pittsburgh. (1980)
20. 20.Holland, J., Reitman, J.: Cognitive Systems Based on Adaptive Algorithms, Pattern-Directed Inference Systems, Academic Press. (1978)
21. Mamdani, E.H.: Application of Fuzzy Algorithms for Control of Simple Dynamic Plant, IEEE Proceedings, Vol. 121, No. 12. (1974) 1585-1588
22. 22.Bäck, T., Hoffmeister, F., Schwefel, H.: A Survey of Evolution Strategies, Proceedings of the Fourth International Conference on Genetic Algorithms. (1991) 2–9
23. 23.Li, J., Liu, H.: Kent Ridge Biomedical Data Set Repository, http://sdmc.i2r.a-star.edu.sg/rp/. (2003)
24. Joachims, T.: Making large-Scale SVM Learning Practical, Advances in Kernel Methods - Support Vector Learning, B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press. (1999)

A BP-SCFG Based Approach for RNA Secondary Structure Prediction with Consecutive Bases Dependency and Their Relative Positions Information

Dandan Song and Zhidong Deng

Department of Computer Science, National Laboratory of Information Science and Technology, Tsinghua University, Beijing 100084, China
sdd00@mails.tsinghua.edu.cn, michael@tsinghua.edu.cn

Abstract. The prediction of RNA secondary structure is a fundamental problem in computational biology. However, in the existing RNA secondary structure prediction approaches, none of them explicitly take the local neighboring bases information into account. That is, when predicting whether a base is paired, only the long range correlation is considered. As a substructure consists of multiple bases, it is affected by consecutive bases dependency and their relative positions in the sequence. In this paper we propose a novel RNA secondary structure prediction approach through a combination of Back Propagation (BP) neural network and statistical calculation with Stochastic Context-Free Grammar (SCFG) approach, in which the consecutive bases dependency and their relative positions information in the sequence are incorporated into the predicting process. When performing on tRNA dataset and three species of rRNA datasets, compared to the SCFG approach alone, our experimental results show that the prediction accuracy is all improved.

1 Introduction

Not just as a passive carrier of genetic information, RNA molecules are also involved in some of the cell's most fundamental processes including catalysis, pre-mRNA splicing and gene expression regulation. Similar to proteins, the function of RNA molecules is mainly determined by their structures. Due to the fact that experimental techniques such as X-ray crystallography and nuclear magnetic resonance (NMR) to obtain the structure data usually require a great deal of time and cost. Compared to the breakthrough advance in the high-throughput sequencing technology, the gap between the exponentially exploding number of nucleic acid sequences and the slowly accumulating number of structures data is expanding.

Using effective computational methods to predict RNA structure from the knowledge of primary sequence can solve this problem. As RNA secondary structure is the backbone of its three-dimensional structure and it is found to be more conservative in evolution, RNA secondary structure prediction is a meaningful and still challenging task.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 506–517, 2007.

There have been many kinds of RNA secondary structure prediction approaches so far. Early in 1978, Nussinov *et al* proposed a maximal base-paring approach that initiated a conversion of RNA secondary structure prediction into an optimal decision problem and used a dynamic programming approach to directly solve it [1]. This research work is undoubtedly of great significance although the resulting prediction accuracy is poor due to its simplicity. Then Zuker developed a minimum free energy(MFE) approach [2]. The earlier version of the MFE approach has been improved and implemented by several commonly-used packages such as Mfold [3] and RNAfold(ViennaRNA Package) [4].

Stochastic approaches are also applied to the problem of RNA secondary structure prediction, e.g., stochastic context-free grammars (SCFG) [5], Bayesian statistical [6], and partition function [7]. Among these approaches, the SCFG approach is preferred due to its simple and suitable description of RNA secondary structures. In addition, heuristic methods based on Hopfield neural network [8], genetic algorithm (GA) [9] and ant colony optimization [10] have also been proposed. However, all these methods still have several limitations. They are always sensitive to parameters, and most importantly, the prediction accuracy of these methods is still far from perfect.

Among above existing computational approaches of RNA secondary structure prediction, the bases are treated separately with long range correlations. None of these methods explicitly consider the local neighboring bases information. Although in the MFE method, some neighboring information is incorporated by the free energy rules, it is much implicit and also too complex. On the other side, as a substructure is composed of multiple bases, consecutive bases dependency and their relative positions information have influence on the type and the stabilization of the structure.

Our approach is a combination of BP neural network and statistical calculation with the SCFG approach. A BP neural network is constructed to incorporate a base's consecutive neighboring bases dependency, and statistical calculation is made to incorporate the base's relative position information. The SCFG approach is then used with such calculated information together with long range correlation information of the bases to predict the optimal secondary structure of the target RNA sequence.

The experiments were done with RNA datasets including tRNA and three species of rRNA sequences. With different training and test samples, our results showed that the prediction accuracy of the proposed method was higher than the BJK grammar model of the SCFG approach, all of which validated the feasibility and effectiveness of our new approach.

2 Method

Firstly, we assume the target RNA sequence has a length n, and the bases in the sequence are indexed by $1, \ldots, n$ from 5'-end to 3'-end. The bases of the sequence are denoted by $x(i)$ $(i = 1, \cdots, n)$.

2.1 Diagram of the Approach

The overall block diagram of our approach is shown in Fig. 1. The sequence is reading sequentially with a sliding window of a fixed size, and a BP neural network is used to calculate the probability $\mathcal{P}_1$ of the central base in the window to be unpaired using its consecutive neihboring bases dependency. Also, another probability $\mathcal{P}_2$ of the base to be unpaired is calculated by performing statistical calculation to incorporate its relative position information. Then the two probabilities are combined into one value. The value is then used in the initialization step of the SCFG process, which calculates the optimal secondary structure through iterative filling and traceback computation.

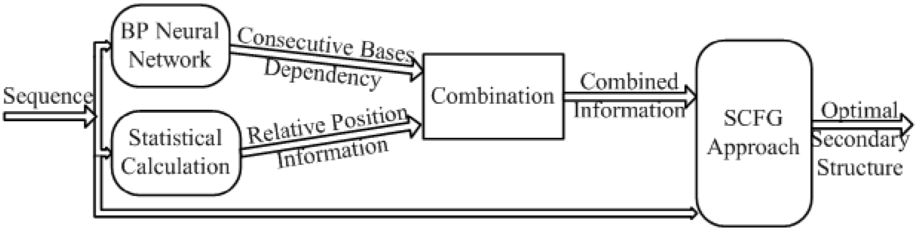


Fig. 1. The overall diagram of our approach

2.2 Consecutive Bases Dependency

A BP neural network is used to incorporate the consecutive bases dependency into the RNA secondary structure prediction. The input of the network is a encoded segment of consecutive bases in the target RNA primary sequence reading by the sliding window, and the output is the probability of the central base in the window to be unpaired. Through training of the neural network, the model is set up and can be used to incorporate consecutive bases dependency of the sequences into RNA secondary structure prediction.

Structure of the BP Neural Network. The structure of the BP neural network is shown in Fig. 2, and is described as follows.

- Input:
  In our experiment, a sliding window of a fixed odd length $2w - 1$ is used to read the primary sequence, denoted as $(W_1, \cdots, W_{w-1}, W_w, W_{w+1}, \cdots, W_{2w-1})$. Currently, w is set to be 7 thus the size of each sliding window is 13 bases, and the overlapping size is set to be 1 base. This means that each sliding window includes 13 bases and goes forward 1 base each time along the RNA sequence. In this case, the subsequence of each window corresponds to one sample.

  In current encoding scheme, each base in each window is expressed by a binary code of 5 bit. Besides the four kinds of RNA bases of ACGU, a new

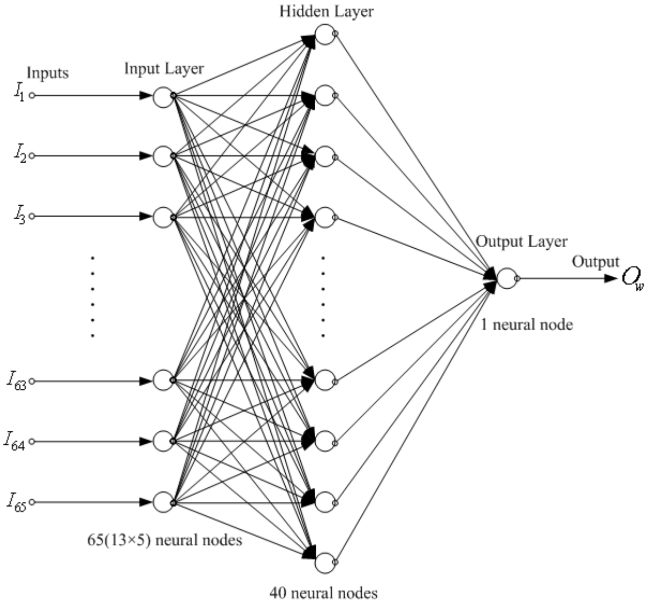


Fig. 2. The structure of the BP neural network

symbol E is defined to denote the end of a sequence, which is added to the two ends of the sequence. The binary codes for each base are:

$$\begin{aligned} A &\Rightarrow 10000 \\ C &\Rightarrow 01000 \\ G &\Rightarrow 00100 \\ U &\Rightarrow 00010 \\ E &\Rightarrow 00001 \end{aligned} \tag{1}$$

Thus for each window, binary sequential codes of $13 * 5 = 65$ bits, which are denoted as $(B_1, B_2, \cdots, B_{64}, B_{65})$, are used to represent the sample. That is,

$$(W_1, W_2, \cdots, W_{13}) \Rightarrow (B_1, B_2, \cdots, B_{65}) \tag{2}$$

An illustration of the sliding window and its binary coding expression is shown in Fig. 3.

Then the input of the BP neural network, $(I_1, I_2, \cdots, I_m)$ where m is the number of the input neural nodes, are set to be the binary codes of each sliding window. Thus, $m = 65$ and

$$(I_1, I_2, \cdots, I_{65}) = (B_1, B_2, \cdots, B_{65}) \tag{3}$$

- Hidden layer:
  Currently, the number of the neural nodes in the hidden layer of the proposed BP neural network model is arbitrarily set to be 40.

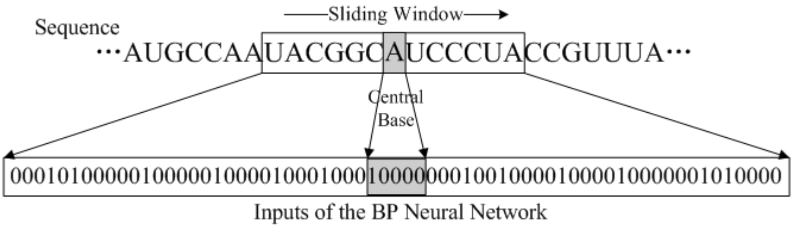


Fig. 3. Illustration of the sliding window and its binary coding expression

- Output:
  For each sliding window, corresponding to its inputs into the BP neural network described above, the one node output O_w of the neural network represents the probability of the central base in the sliding window to be unpaired.

Training. Training samples are RNA sequences with their known secondary structures. For a sequence of length n, its training samples are produced according to the following rules:

(1) Extend the sequence.
Every base of index i ($i = 1, \cdots, n$) should be the center of one sliding window, so to realize this, the sequence is extended by labels E in the two ends. The number of E in each end is $w - 1$, that is, set $x_k = E$ for $k = -(w-2), \cdots, 0$ and $k = n+1, \cdots, n+w-1$.
(2) Encode the bases of the sequence.
The extended sequence is encoded according to (1).
(3) Get inputs and outputs.
Slide the sliding window along the encoded sequence to get inputs. That is, the subsequence read by the sliding window is

$$\begin{gathered}(W_1, W_2, \cdots, W_7, \cdots, W_{12}, W_{13}) \\ = (x_{i-(w-1)}, x_{i-w}, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_{i+w-2}, x_{i+(w-1)}) \\ (i = 1, \cdots, n).\end{gathered} \tag{4}$$

which is encoded by (2) and input into the neural network as (3).

The corresponding output O_w is the known structure status of the central base x_i. If it is unpaired, then set $O_w = 1$, otherwise $O_w = 0$.

After the training samples are constructed, then the traditional BP learning algorithm [11] is used to train the parameters.

Inference. For the target sequence to be predicted, inputs are attained in the same way as the training sequences. The output of the neural network O_w is a value in the interval of $[0, 1]$, which represents the probability $\mathcal{P}_1(i) = O_w$ for the central base x_i to be unpaired.

2.3 Relative Position Information

Training. The position of a base is assumed to have influence on its probability to be unpaired. Since the lengths of the RNA sequences are different, relative positions of their bases are used. In the training process, the positions of unpaired bases relative to their whole sequences $rp^i_{n'} = i/n'$ are stored, where $rp^i_{n'}$ is the relative position of the unpaired base of index i in the sequence of length n' in the training dataset.

Inference. For the target sequence, when taking the relative positions information of the bases into account, the computation of the probabilities of the bases to be unpaired is as follows.

After relative positions of the unpaired bases in the training samples are stored in the training process, they are multiplied by the length n of the target sequence. Then absolute positions are attained in a statistical sense, denoted as

$$rp^i_{n'} \times n \tag{5}$$

By rounding above results to the nearest integers, these absolute positions are scattered into the base positions of the target sequence. Then for positions $i = 1, \ldots, n$ of the target sequence, discrete frequencies $\mathcal{P}_i$ of these elements occurred in the training dataset are computed using Laplace prior (plus-one) equation given in (6).

$$\mathcal{P}_i = \frac{N_i + 1}{N + I}, \ (i = 1, 2, \cdots, I, \ N = \sum N_i). \tag{6}$$

where N_i denotes the occurrence number of the ith case and I is the number of cases. Therefore, when taking account of relative positions information, the frequencies of the base $x(i)$ to be unpaired $\mathcal{P}_2$ are specified by:

$$\mathcal{P}_2(i) = \frac{N_i + 1}{\sum_{j=1}^{n} N_j + n} \tag{7}$$

where $i = 1, \ldots, n$, N_i and N_j denote the occurrence numbers of the unpaired bases in positions i and j, respectively.

2.4 Optimal Structure Calculation

The SCFG approach is used to calculate the optimal secondary structure with the maximum likelihood. As [12] pointed out, among nine SCFG models, the performance of Knudsen/Hein's BJK grammar [13] is only slightly lower than its extending to $G6^S$ grammar with a first order Markov chain. While including stacking parameters makes the $G6^S$ grammar much more complex, in the synthetic sense, BJK grammar can be taken as the best SCFG grammar. Therefore, the BJK grammar of the SCFG approach is used here.

The production rules of the BJK grammar of SCFG approach are:

$$\begin{aligned} S &\rightarrow LS \mid L \\ L &\rightarrow aFa' \mid a \\ F &\rightarrow aFa' \mid LS \end{aligned}$$

where S L and F are nonterminals, a and a' are terminals which represent bases.

Estimation of the SCFG Parameters. The parameters of the SCFG approach is simply estimated by their frequencies occurred in the training samples with Laplace (plus-one) prior function given in (6). Although it is not accurate as the inside-outside learning algorithm [14], it can mainly satisfy current requirement and the computational complexity is much lower.

For the production rules that produce bases, such as $L \rightarrow aFa'$ $L \rightarrow a$ and $F \rightarrow aFa'$, the probability of the production rule can be divided into two parts. That is, the probability of the transition from the left of the rule to the right, and the probability of the emission of the specific base or base pairs. By multiplying these two probabilities, the result value equals to the probability of the production rule.

- Probabilities of Transition Rules
  The probabilities of the transition rules in the BJK grammar of SCFG approach are computed as

$$\mathcal{P}_{tran}(R_i^r) = \frac{N_i^r + 1}{\sum_j N_j^r + N^r} \tag{8}$$

  where R_i^r denotes the ith transition rule of nonterminal r, N_i^r denotes its occurrence number, and N^r is the number of the transition rules of nonterminal r. For the specific rules of the BJK grammar of SCFG, $N^r = 2$ for $r = \{L, S, F\}$.
- Probabilities of Emission Rules
  After the occurrence numbers of base pairs and single bases in the annotated RNA secondary structures are counted, the probabilities of the emission rules emitting base pairs and single bases are determined by (9) and (10).

$$\mathcal{P}_{pair}(X,Y) = \frac{N_{X,Y} + 1}{\sum_{X,Y} N_{X,Y} + 16} \quad X,Y \in \{A,C,G,U\} \tag{9}$$

$$\mathcal{P}_{single}(X) = \frac{N_X + 1}{\sum_X N_X + 4} \quad X \in \{A,C,G,U\} \tag{10}$$

  where $\mathcal{P}_{pair}(X,Y)$ denotes the probability to emit the base pair (X,Y), and $N_{X,Y}$ is its occurrence number in the training samples. Similarly, $\mathcal{P}_{single}(X)$ and N_X denote the probability to emit the single base X and its occurrence number respectively.

Combination of the Two Kinds of Probabilities
(1) Scale the two kinds of unpaired probabilities for bases in the target sequence, making the maximum value to be 1:

$$\mathcal{P}_h(i) = \frac{\mathcal{P}_h(i)}{\max_{1 \le j \le n} \mathcal{P}_h(j)} \ (h = 1, 2) \tag{11}$$

(2) Combine the probabilities of the base to be unpaired with its consecutive bases dependency information and its relative position information:

$$\mathcal{P}(i) = \frac{\mathcal{P}_1(i) + \mathcal{P}_2(i)}{2} \tag{12}$$

(3) Scale the combined probabilities making the maximum value to be 1:

$$\mathcal{P}(i) = \frac{\mathcal{P}(i)}{\max_{1 \le j \le n} \mathcal{P}(j)} \tag{13}$$

Prediction of the Secondary Structure. The secondary structure with the maximum likelihood is computed by the dynamic programming algorithm. The symbols $L(i,j)$ $S(i,j)$ and $F(i,j)$ denote that, when bases $x(i)$ and $x(j)$ construct the specific structure under the meanings of nonterminals L S and F, the maximum likelihood of the substructure composing of the bases between indices i and j $(x(i), \cdots, x(j))$. The calculation of these values initiates from subsequences of length 1 and expands to longer subsequences through iteration. In our approach, the initialization step is adjusted to incorporate the probabilities of the bases to be unpaired which has been calculated with consecutive bases dependency and relative positions information. It is given in (14).
Initialization:

For $L(i,j)$ when $i = j$,

$$\mathcal{P}(L(i,i)) = \mathcal{P}_{single}(x(i)) * \mathcal{P}_{tran}(L \to a) * \underline{(\mathcal{P}(i))} \tag{14}$$

Other iterative calculation is the same as in the Viterbi process of the SCFG approach, which is specified in the following equations.

(1)$S(i,j)$,when $i \le j$,

$$\mathcal{P}(S(i,j)) = \max \begin{cases} \max\limits_{i \le k < j} \mathcal{P}(L(i,k)) * \mathcal{P}(S(k+1,j)) * \mathcal{P}_{tran}(S \to LS) \\ \mathcal{P}(L(i,j)) * \mathcal{P}_{tran}(S \to L) \end{cases} \tag{15}$$

(2)$F(i,j)$, when $i \le j$,

$$\mathcal{P}(F(i,j)) = \max \begin{cases} \max\limits_{i \le k < j} \mathcal{P}(L(i,k)) * \mathcal{P}(S(k+1,j)) * \mathcal{P}_{tran}(F \to LS) \\ \mathcal{P}(F(i+1,j-1)) * \mathcal{P}_{pair}(x(i),x(j)) * \mathcal{P}_{tran}(F \to aFa') \end{cases} \tag{16}$$

(3)$L(i,j)$,when $i < j$,

$$\mathcal{P}(L(i,j)) = \mathcal{P}(F(i+1,j-1)) * \mathcal{P}_{pair}(x(i),x(j)) * \mathcal{P}_{tran}(L \rightarrow aFa') \qquad (17)$$

Using the iteration process, the maximum likelihood of the optimal structure is calculated, then traceback process is used to trace the corresponding secondary structure.

3 Experimental Results

3.1 Dataset Preparation

In experiments, a tRNA dataset and three species of rRNA datasets are used. In the tRNA dataset, there are 843 tRNA sequences with annotated secondary structures taken from the EMBL databank [15], including various series such as virus, archaea, eubacteria, cyanelle, cytoplasm and mitochondria. Three training datasets are constructed. The first one is named MT10CY10, whose 10 tRNA sequences are randomly selected from the cytoplasm data and 10 sequences from the mitochondria data; while the second one is MT100, whose 100 tRNA sequences are randomly selected from the mitochondria data. And the Rand tRNA dataset is composed of 569 randomly selected tRNA from all the series.

In the rRNA datasets, three different species of rRNA are used: SRP (Singal Recognition Particle) dataset [16], tmRNA dataset [17] and RNaseP dataset [18]. In these datasets, there are 81 rRNA sequences in the SRP dataset, 97 sequences in the tmRNA dataset and 225 sequences in the RNaseP dataset with their known secondary structures. For the former two datasets, both 40 sequences are randomly selected to form training samples while the others form test samples; For the RNaseP dataset, 100 sequences are randomly selected to form training samples and the others form test samples. And in each experiment, different random selection are performed to form new training and test samples.

Table 1. The Sensitivity(%) value of the proposed CBRP approach compared with the BJK grammar of the SCFG approach with the same training (columns) and test (rows) datasets. For instance, the first number 83.07 refers to the prediction accuracy of the BJK grammar of SCFG using the MT10CY10 dataset to train parameters and using the ARCHAE dataset to test.

	MT10CY10		MT100		Rand tRNA	
Dataset	SCFG	CBRP	SCFG	CBRP	SCFG	CBRP
ARCHAE	83.07	89.14	81.16	87.91	83.26	93.25
CY	81.88	91.39	81.59	91.17	83.01	92.09
CYANELCHLORO	84.22	87.07	83.44	89.41	85.71	91.75
EUBACT	90.06	88.85	87.56	89.74	91.19	93.34
VIRUS	81.28	88.68	79.01	86.21	80.45	88.68
MT	77.88	81.42	78.56	86.81	73.47	79.58
PARTIII	77.11	74.36	77.39	81.36	73.98	77.11

Table 2. The Specificity(%) value of the proposed CBRP approach compared with the BJK grammar of the SCFG approach

	MT10CY10		MT100		Rand tRNA	
Dataset	SCFG	CBRP	SCFG	CBRP	SCFG	CBRP
ARCHAE	75.40	91.61	73.97	90.17	77.99	97.66
CY	78.03	93.72	77.19	92.32	82.65	97.69
CYANELCHLORO	79.88	91.02	78.54	93.42	83.75	97.04
EUBACT	84.05	89.37	81.73	90.78	86.84	96.54
VIRUS	78.69	90.93	74.13	90.50	79.80	94.52
MT	76.57	89.92	76.90	93.64	78.28	94.39
PARTIII	75.74	86.00	76.88	91.49	78.04	93.79

Table 3. For the dataset of SRP rRNA, the prediction accuracy compared between the proposed CBRP approach and the BJK grammar of SCFG implemented in [12]

	Sensitivity(%)		Specificity(%)	
Test No.	SCFG	CBRP	SCFG	CBRP
1	55.46	59.96	50.26	56.67
2	50.62	56.77	45.55	53.13
3	54.68	62.34	49.87	60.47
4	58.34	63.27	54.14	61.83
5	55.40	59.52	50.36	56.97
6	57.34	62.67	54.49	60.49
7	57.70	62.93	52.51	59.08

3.2 Comparison to the BJK Grammar of the SCFG Approach

Using the same training and test samples, our results are compared with the BJK grammar model of the SCFG approach. *Sensitivity* and *Specificity* parameters are used to evaluate the RNA secondary structure prediction accuracy, which are common measures of the accuracy of prediction approaches. The *Sensitivity* value denotes the percentage of base-pairs that are predicted correctly among actual base-pairs in the reference structure, while the *Specificity* value

Table 4. For the dataset of tmRNA rRNA, the prediction accuracy compared between the proposed CBRP approach and the BJK grammar of SCFG implemented in [12]

	Sensitivity(%)		Specificity(%)	
Test No.	SCFG	CBRP	SCFG	CBRP
1	39.44	46.03	37.68	52.73
2	37.37	44.02	34.37	48.89
3	39.72	41.83	37.54	47.64
4	37.48	44.49	35.74	50.44
5	38.69	45.6	36.11	49.7
6	38.59	42.21	36.35	47.04
7	39.54	45.69	38.27	50.59

Table 5. For the dataset of RNaseP rRNA, the prediction accuracy compared between the proposed CBRP approach and the BJK grammar of SCFG implemented in [12]

	Sensitivity(%)		Specificity(%)	
Test No.	SCFG	CBRP	SCFG	CBRP
1	46.21	53.57	45.90	55.74
2	47.48	56.37	44.93	55.20
3	48.62	56.66	46.56	56.42
4	48.40	58.58	45.97	57.90
5	47.13	58.62	43.57	56.08
6	46.35	56.01	43.72	54.99
7	44.97	57.17	45.00	58.34

denotes the percentage of base-pairs that are predicted correctly among predicted base-pairs. Tab. 1 and 2 express the prediction accuracy of our approach (CBRP) compared with the BJK grammar of the SCFG approach (SCFG) using the tRNA database. The results show that our method outperforms the BJK grammar of the SCFG approach significantly.

For the SRP dataset of rRNA, 7 experiments are done with different training and test samples which are selected randomly. The prediction accuracy of our approach and the SCFG approach is shown in Tab. 3, our approach also outperforms the SCFG approach.

The Tab. 4 and Tab. 5 show the prediction results of our approach compared with the SCFG approach when performing on tmRNA and RNaseP rRNA datasets with 7 different training and test datasets selected randomly. The prediction accuracy is all greatly improved by our approach.

4 Conclusion and Future Work

By combining a BP neural network and statistical calculation with the SCFG approach, our approach incorporates consecutive bases dependency and their relative positions information into the RNA secondary structure prediction process. With the same training and test samples randomly selected from four kinds of RNA datasets including tRNA and rRNA, our experimental results are compared with the SCFG approach alone. The greatly improved prediction accuracy demonstrates the effectiveness of our approach. And the future work includes:

- The size of the sliding window is arbitrarily set to be 13 by taking reference of the proteins secondary structure prediction. But as the characters of RNA and protein are different, the size should be fixed through experiments.
- The number of neural nodes in the hidden layer of the BP neural network is also arbitrarily set, it can be adjusted to better suit features of the samples.
- Traditional BP neural network and the learning algorithm are used. As other revised network structure and much more efficient learning algorithms are prompted in recent years, the neural network can be improved.
- The proposed approach has only been performed in the training and test samples selected from the same species of RNA. Experiments between

different species of datasets should be performed. And it needs to be pointed out that when no training sets are available then energy minimization methods such as the Mfold/Vienna are the best that can be tried.

Acknowledgement

This work was supported in part by the National Science Foundation (Grant No. 60621062) and the Teaching and Research Award Program for Outstanding Young Teachers in Higher Education Institutions of MOE (TRAPOYT), China.

References

1. Nussinov, Pieczenik, Griggs, Kleitman: Algorithms for loop matchings. SIJAM: SIAM Journal on Applied Mathematics **35** (1978) 68–82
2. Zuker, M., Stiegler, P.: Optimal computer folding of large rna sequences using thermodynamics and auxiliary information. Nucleic Acids Research **9**(1) (1981) 133–148
3. Zuker, M.: Mfold web server for nucleic acid folding and hybridization prediction. Nucleic Acids Research **31**(13) (2003) 3406–3415
4. Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, L.S., Tacker, M., Schuster, P.: Fast folding and comparison of RNA secondary structures. Monatsh. Chem. **125** (1994) 167–188
5. Eddy, S.R., Durbin, R.: Rna sequence analysis using covariance models. Nucleic Acids Res **22**(11) (1994) 2079–2088
6. Ding, Y.: Statistical and bayesian approaches to rna secondary structure prediction. RNA **12** (2006) 323–331
7. Mccaskill, J.S.: The equilibrium partition function and base pair binding probabilities for rna secondary structure. Biopolymers **29**(6-7) (1990) 1105–1119
8. Steeg, E.W.: Neural network algorithms for rna secondary structure prediction. Technical report, University of Toronto Computer Science Dept., Toronto Canada (1990)
9. Hu, Y.J.: Gprm: a genetic programming approach to finding common rna secondary structure elements. Nucleic Acids Research **31**(13) (2003) 3446–3449
10. McMillan, N.: RNA secondary structure prediction using ant colony optimization. Master's thesis, School of Informatics, The University of Edinburgh (2006)
11. Rumelhart, D., Hinton, G., Williams, R.: Learning representations by back-propagating errors. Nature **323** (1986) 533–536
12. Dowell, R.D., Eddy, S.R.: Evaluation of several lightweight stochastic context-free grammars for rna secondary structure prediction. BMC Bioinformatics **5** (2004) 71–99
13. Knudsen, B., Hein, J.: Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. Bioinformatics **15**(6) (1999) 446–454
14. Lari, K., Young, S.J.: The estimation of stochastic context-free grammars using the inside-outside algorithm. Computer Speech and Language **4** (1990) 35–56
15. Steinberg, S., Misch, A., Sprinzl, M.: Compilation of tRNA sequences and sequences of tRNA genes. Nucleic Acids Research **21**(13) (1993) 3011–3015
16. Rosenblad, M.A., Gorodkin, J., Knudsen, B., Zwieb, C., Samuelsson, T.: SRPDB: Signal recognition particle database. Nucleic Acids Research **31**(1) (2003) 363–364
17. Zwieb, C., Gorodkin, J., Knudsen, B., Burks, J., Wower, J.: tmrdb (tmrna database). Nucleic Acids Research **31**(1) (2003) 446–447
18. Brown, J.W.: The ribonuclease P database. Nucleic Acids Research **27**(1) (1999) 314

Delta: A Toolset for the Structural Analysis of Biological Sequences on a 3D Triangular Lattice*

Minghui Jiang**, Martin Mayne, and Joel Gillespie

Department of Computer Science, Utah State University,
Logan, Utah 84322-4205, USA
mjiang@cc.usu.edu

Abstract. The lattice approach to biological structural analysis was made popular by the HP model for protein folding, but had not been used previously for RNA secondary structure prediction. We introduce the Delta toolset for the structural analysis of biological sequences on a 3D triangular lattice. The Delta toolset includes a proof-of-concept RNA folding program that is both fast and accurate in predicting the secondary structures with pseudoknots of short RNA sequences.

1 Introduction

The prediction and analysis of the folded structures of biological sequences such as proteins and RNAs are important research problems in bioinformatics and computational biology. Protein folding, for example, is such a difficult problem that it has been studied extensively in various simplified models. In the HP model [10,11], the 20 types of amino acids are grouped to two types, hydrophobic (H) and hydrophilic (P). The folding takes place on a lattice: each amino acid occupies a unique lattice point; consecutive amino acids in the protein sequence occupy adjacent lattice points. Given a protein sequence, the problem reduces to finding a lattice folding of the sequence with the maximum number of H-H contacts.

The prediction of RNA secondary structures with pseudoknots is another difficult problem in structural bioinformatics. We review some basic concepts. The *primary structure* of an RNA is the sequence of nucleotides (that is, the four different bases A, C, G, and U) in its single-stranded polymer. An RNA folds into a three-dimensional structure by forming hydrogen bonds between pairs of complementary bases, such as the Watson-Crick pairs G-C and A-U and the wobble pair G-U, which are nonconsecutive in the sequence. The three-dimensional arrangement of the atoms in the folded RNA molecule is its *tertiary structure*; the collection of base pairs is its *secondary structure*. A *pseudoknot* in an RNA secondary structure is composed by two interleaving base pairs (i, j) and (k, l) with sequence indices $i < k < j < l$.

Most early research on RNA secondary structure prediction adopts the thermodynamic approach: each candidate secondary structure corresponds to a global free energy, which depends on the recursive decomposition of the structure and on a set of experimentally determined energy parameters; the optimal secondary structure has the

* Supported by Utah State University research funds A13501 and A14766.

** Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 518–529, 2007.

minimum global free energy. When pseudoknots are excluded, the optimal secondary structure of an RNA can be effectively computed by dynamic programming algorithms in $O(n^3)$ time and $O(n^2)$ space [27,36,15,24]. There has been considerable effort [28,33,3] on extending the dynamic programming algorithms to include pseudoknots in RNA secondary structures. However, these extended algorithms typically have very high complexities, ranging from $O(n^4)$ to $O(n^6)$ in time and from $O(n^3)$ to $O(n^4)$ in space, which make them impractical even for RNA sequences of only a few hundred bases. Furthermore, as noted by Lyngsø and Pedersen [23] and also by Ieong et al. [16], these algorithms can handle only limited types of pseudoknots: the exact types are implicit in the algorithms and difficult to determine. On the other hand, if arbitrary pseudoknots may be included in the secondary structures, then the prediction problem becomes exceedingly difficult. Lyngsø and Pedersen [23] showed that the problem of determining the optimal secondary structure possibly with pseudoknots is NP-hard even for a simple nearest-neighbor energy model. Similar hardness results have been shown for other models that permit arbitrary pseudoknots [3,16,34,22]. Although approximation algorithms [16,22,8,17,18] have been proposed for optimization problems in these models, the practical relevance of these algorithms is diminished by either the large approximation ratios or the (still) high time and space complexities. Alternative methods, such as maximum weighted matching [31] and iterated loop matching [29], are based on a combination of thermodynamic and comparative approaches. These methods can often achieve better prediction accuracies than algorithms based on the thermodynamic approach alone, but the necessary information for comparative studies (for example, homologous sequences) are not always available.

The apparent difficulty of the prediction of RNA secondary structures with pseudoknots naturally prompts researchers to explore heuristic approaches such as Monte-Carlo simulation [1] and genetic algorithms [13,4,30]. However, to the best of our knowledge, no previous work has been done for this problem using the lattice approach. This is somewhat surprising, especially in light of the fact that the related problem of protein folding has been extensively studied in the HP model [14,2,6,7,25,26,21,19]. To evaluate the feasibility of the lattice approach to RNA secondary structure prediction, we have developed the Delta toolset for the structure analysis of biological sequences on a 3D triangular lattice. The toolset includes a proof-of-concept RNA folding program based on simulated annealing [20] with pull moves [21], which is shown by our experiment to be both fast and accurate in predicting the secondary structures with pseudoknots of short RNA sequences. The classic two-step approach to RNA structure prediction, which first predicts the secondary structure then derives the compatible tertiary structure, can be problematic since a lot of secondary structures that are valid under the typical constraints of the existing models (for example, no base is included in more than one base pair, and the bases of any base pair obey the minimum separation requirement) cannot be realized as three-dimensional structures due to steric constraints. Our lattice approach avoids this problem by simulating the RNA tertiary structure on a 3D triangular lattice directly; the secondary structure is then derived from the tertiary structure, not vice versa.

The rest of the paper is organized as follows. In Section 2, we introduce the Delta toolset. In Section 3, we present our experimental results on the effectiveness of our

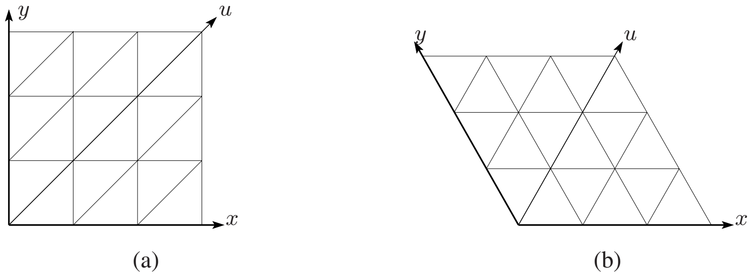
RNA folding program in predicting RNA secondary structures with pseudoknots. We conclude in Section 4.

2 The Delta Toolset

The Delta toolset[1] is implemented in the C programming language. It consists of three components: the Delta library, which includes the core data structures, input/output utilities, programming interfaces, and biology-specific information; and two front-end tools, `show` and `fold`, for the visualization and manipulation of the folded structures of biological sequences on a 3D triangular lattice.

2.1 The 3D Triangular Lattice

We briefly examine the 2D triangular lattice first. We refer to Fig. 1. Let $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively, be the unit vectors along the x and y axes of a square lattice (a). Add an auxiliary axis $\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{y}$ along the xy diagonal of the square lattice, then skew the lattice until the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ becomes $2\pi/3$, and we obtain a 2D triangular lattice (b). To specify a lattice point on a 2D triangular lattice, the coordinates on the two primary axes along $\boldsymbol{x}$ and $\boldsymbol{y}$ are sufficient. For example, each lattice point $\boldsymbol{p} = (x, y)$ has six neighbors $(x + 1, y)$, $(x - 1, y)$, $(x, y + 1)$, $(x, y - 1)$, $(x + 1, y + 1)$, and $(x - 1, y - 1)$. With the auxiliary axis $\boldsymbol{u}$, the six neighbors of $\boldsymbol{p}$ can be more succinctly specified by $\boldsymbol{p} \pm \boldsymbol{x}$, $\boldsymbol{p} \pm \boldsymbol{y}$, and $\boldsymbol{p} \pm \boldsymbol{u}$.



(a) (b)

Fig. 1. The 2D triangular lattice

We now consider the 3D triangular lattice formed by a closest cubic packing of spheres. We refer to Fig. 2. An interesting property of the 3D triangular lattice is that it consists of layers of *square* lattices in parallel planes. Beside the two unit vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ that specify the two primary axes of each square lattice, the 3D triangular lattice has an additional primary axis along a unit vector $\boldsymbol{z}$, which is at an angle of $2\pi/3$ from both $\boldsymbol{x}$ and $\boldsymbol{y}$. The distance between two adjacent planes, which support two consecutive layers of square lattices, is $\sqrt{2}/2$. Each lattice point on a 3D triangular lattice has 12 neighbors, distributed among three consecutive layers of square lattices, four on each layer.

[1] `http://www.cs.usu.edu/~mjiang/delta/`

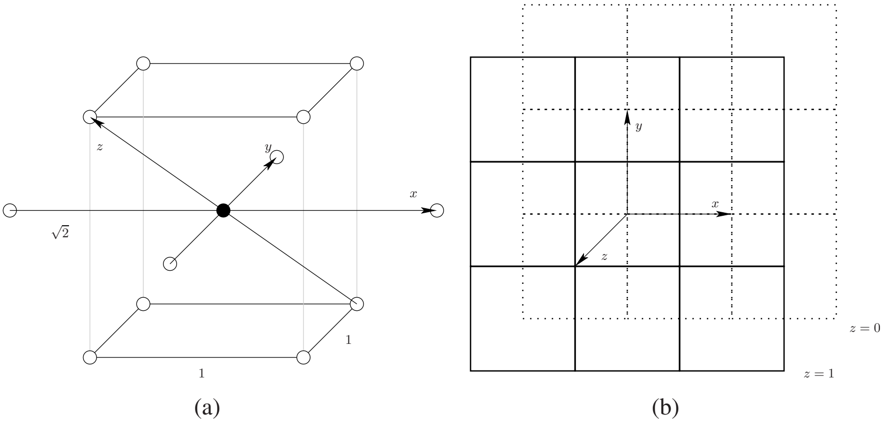


Fig. 2. The 3D triangular lattice. (a) Each lattice point has 12 neighbors distributed in three square lattices in parallel planes. (b) A projected view: alternating layers of square lattices are depicted by solid and dotted lines.

In a 3D triangular lattice, the location of each lattice point can be specified by its coordinates on the three primary axes along $\boldsymbol{x}$, $\boldsymbol{y}$, and $\boldsymbol{z}$. Since the three axes are not orthogonal, the Cartesian coordinates (x', y', z') of a lattice point $\boldsymbol{p} = (x, y, z)$ need to be computed from the following equations:

$$x' = x - z/2, \qquad y' = y - z/2, \qquad z' = z/\sqrt{2}.$$

For convenience, we also define three auxiliary axes along the three unit vectors $\boldsymbol{u} = \boldsymbol{x} + \boldsymbol{z}$, $\boldsymbol{v} = \boldsymbol{y} + \boldsymbol{z}$, and $\boldsymbol{w} = \boldsymbol{x} + \boldsymbol{y} + \boldsymbol{z}$. The 12 neighbors of a lattice point $\boldsymbol{p}$ are $\boldsymbol{p} \pm \boldsymbol{x}$, $\boldsymbol{p} \pm \boldsymbol{y}$, $\boldsymbol{p} \pm \boldsymbol{z}$, $\boldsymbol{p} \pm \boldsymbol{u}$, $\boldsymbol{p} \pm \boldsymbol{v}$, and $\boldsymbol{p} \pm \boldsymbol{w}$; each neighbor can be uniquely indexed by a number between 1 and 12.

We use the 3D triangular lattice instead of the more common cubic lattice for two reasons: first, the 3D triangular lattice is "denser" than the cubic lattice and therefore approximates the 3D space better; second, the 3D triangular lattice is not susceptible to the parity problem [2] associated with the cubic lattice, in which two elements with the same parity in the sequence cannot be adjacent in the lattice.

2.2 Representation of Structures

On the 3D triangular lattice, the folded structure of a biological sequence S of n elements can be represented by a *turn sequence* T of $n - 1$ directions that simulates a self-avoiding walk on the lattice. The directions in the turn sequence are over the alphabet $\{$`U,V,W,X,Y,Z,u,v,w,x,y,z`$\}$; the upper and the lower cases of each letter, respectively, correspond to the positive and the negative directions along the corresponding axis.

The *configuration* of a biological sequence on the 3D triangular lattice is its 3D structure, represented in the Delta library as an array of lattice points occupied by the elements in the sequence. As the Delta library reads the turn sequence in, an initial configuration is "decoded": place the first element at the origin, then repeatedly follow the next direction in the turn sequence to one of the 12 neighbors of the current lattice point and place the next element there. As the configuration is decoded, each element is also inserted into a hashtable with its lattice coordinates as the key. When the configuration changes during the structural manipulation, both the coordinates array and the hashtable are updated accordingly. The configuration is *valid* if no two elements occupy the same lattice point and if consecutive elements in the sequence occupy adjacent lattice points. The hashtable is used so that collisions can be detected very efficiently when maintaining the validity of the configuration.

To export the structure of the biological sequence, the Delta library "encodes" the coordinates array back into a turn sequence by comparing the differences in the coordinates of consecutive elements. A biological sequence and its lattice configuration can be stored externally in a text file that contains a sequence of elements over the alphabet {`A-Za-z`} and a sequence of turns over the alphabet {`U-Zu-z`}.

2.3 Visualization of Structures

The Delta toolset includes a visualizer, called `show`, that can display the structure of a biological sequence on a 3D triangular lattice. The `show` program is implemented using the Delta library and the OpenGL graphics library.

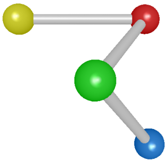


Fig. 3. An example sequence displayed by the visualizer

Given the input consisting of a sequence of elements and a sequence of turns, the visualizer decodes the lattice coordinates of the elements from the turn sequence using the Delta library, computes the Cartesian coordinates of the occupied lattice points, then displays the 3D structure of the biological sequence in the *ball-and-stick* model: each element is represented by a ball; two consecutive elements are connected by a stick. Elements of different types (specified by different letters) are rendered by the visualizer in different colors. For example, with a biological sequence `CAGT` and its turn sequence `XZv` saved in a file `input.txt`, the command `show -i input.txt` displays the structure in Fig. 3. The visualizer also provides a `-rna` option that enables the display of base pairings in the RNA secondary structure. When the 3D structure of a sequence is displayed, the user can use the mouse and the keyboard to rotate, zoom, and adjust the view. All figures of RNA structures included in this paper are processed from actual screenshots of the visualizer.

2.4 Manipulation of Structures by Pull Moves

The Delta library provides programming interfaces for the structural manipulation of biological sequences by pull moves. The pull moves were introduced by Lesh et al. [21] as a complete and effective move set for protein folding on the square lattice in the HP model; it was later adapted to the hexagonal lattice by Jiang and Zhu [19]. We adapt the pull moves to the 3D triangular lattice.

For a sequence S, denote by $S[i]$ the i'th element of S. Given a lattice configuration of a sequence S of n elements, for each element $S[i]$, $1 \leq i \leq n$, and for each direction $d \in \{\texttt{U-Zu-z}\}$, there is a possible pull move $pull(i, d)$. Denote by $p(i, d)$ the neighbor of (the lattice point adjacent to) $S[i]$ in the direction d. The element $S[i+1]$ or $S[i-1]$ is called an *anchor* of the pull move $pull(i, d)$ if it is also adjacent to $p(i, d)$. For convenience, define two imaginary elements $S[0]$ and $S[n+1]$ such that $S[0]$ is always an anchor of $S[1]$ and that $S[n+1]$ is always an anchor of $S[n]$. A pull move $pull(i, d)$ is *valid* if $p(i, d)$ is unoccupied and the pull move has at least one anchor.

Let $pull(i, d)$ be a valid pull move on a valid configuration. To execute the pull move, first move $S[i]$ to $p(i, d)$. If the pull move has two anchors $S[i-1]$ and $S[i+1]$, then this move of $S[i]$ alone, which is called a *flip*, already results in a valid configuration, and the pull move is finished. If the pull move has only a single anchor, say, $S[i-1]$, then continue to "pull" $S[k]$ to the former location of $S[k-1]$, for $k = i+1, \ldots, n$, until $S[k]$ is adjacent to the new location of $S[k-1]$. The other case, that $S[i+1]$ is the only anchor, is symmetric. A valid pull move clearly maintains the two properties that no two elements occupy the same lattice point and that consecutive elements in the sequence occupy adjacent lattice points. The resulting configuration is still valid.

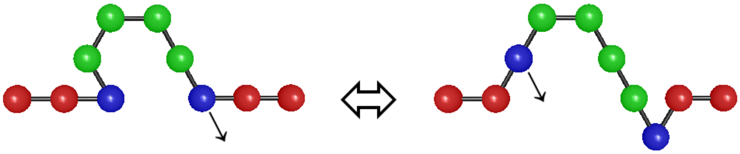


Fig. 4. An example of a pull move

We refer to Fig. 4 for an example of a pull move. Moving the element as indicated changes the sequence from one configuration to the other. The unmoved element next to the first moved element in the sequence is the anchor. An important property of the pull moves is that they are *reversible*. Each pull move results in a contiguous range of elements being relocated. To return a sequence to its old configuration, another pull move is sufficient: simply move the last element in the range back to its former location; the remaining elements in the range are then "pulled" one by one to their respective former locations. Another property of the pull moves is that they are *complete*, that is, any valid configuration can be moved to any other valid configuration by a sequence of pull moves. It is easy to check that any configuration v can be pulled into a straight line l by a sequence of pull moves $move(v, l)$. Since each pull move is reversible, a move sequence $move(u, v)$ can be composed by concatenating the move sequence $move(u, l)$ and the reversed move sequence $move(l, v)$.

As noted by Lesh et al. [21], an important aspect of pull moves in terms of their effectiveness on protein folding is that they make relative small adjustments to a given configuration, or, more accurately, the average number of relocated elements of a pull move is small. To test whether this "small adjustment" property is preserved in our adaptation of pull moves to the 3D triangular lattice, we ran random pull moves on sequences of various sizes to measure the numbers of relocated elements of the pull moves. Let ℓ be the length of a test sequence. Our test program initializes the sequence in a straight-line configuration, executes $10^5\ell$ random pull moves as warm-up, then executes another $10^5\ell$ random pull moves to gather statistics. Each random pull move $pull(i, d)$ is specified by two uniformly random numbers: a number between 1 and ℓ for the index i, and another number between 1 and 12 for the direction to move $S[i]$. If a random pull move is valid, it is committed and the number of relocated elements is recorded; otherwise, the move is aborted and the configuration is reverted.

Table 1. Statistics on the number of relocated elements of random pull moves

sequence size	32	64	128	256	512
average	2.40625	3.79688	4.10938	3.36328	3.48438
standard deviation	1.29164	2.65581	3.21716	1.97754	2.40347

We tested sequence sizes ranging from 32 to 512, typical of biological sequences such as RNAs and proteins. As we can see from the results listed in Table 1, the number of relocated elements of a pull move is on average a small constant.

2.5 The Structural Manipulation Tool

The Delta toolset includes a structural manipulation tool called `fold`, which allows a user to input a sequence, manipulate it with pull moves, then output the resulting structure. We introduce two important command-line options of `fold`:

`-z` n performs n random (warm-up) pull moves on the sequence.

`-a` n n' k performs simulated annealing for k rounds; each round attempts at most n random pull moves and aborts if the configuration does not improve after n' consecutive moves (the two parameters n' and k are optional; by default, $n' = n$ and $k = 1$).

These options allow the use of `fold` as an RNA folding program based on simulated annealing with pull moves. Each configuration v of a sequence has a corresponding score $s(v) \geq 0$. With the sequence set to an arbitrary configuration v_0 at the beginning (either directly from the input, or after a number of random warm-up moves specified by the `-z` option), the RNA folding program performs simulated annealing for n steps (specified by the `-a` option). At the i'th step, a random pull move is chosen, which may transform the old configuration v_{i-1} into a new configuration v_i. If $s(v_i) \geq s(v_{i-1})$, the pull move is committed and the simulated annealing procedure continues to the next step; otherwise, the pull move is committed with a probability of $p_i = 2^{(s(v_i)-s(v_{i-1}))/T_i}$, where $T_i = c/\log_2(1+i/n)$ is the system temperature at the i'th step. This cooling schedule is chosen because of its special statistical properties [35]. When $i = 0$, we have $T_0 = \infty$; when $i = n$, we have $T_n = c$. Intuitively, the system can be found in any configuration with approximately equal probabilities at a high temperature, and is more likely to be found in a configuration with a high score at a low temperature. The constant c is set to $1/\log_2 10$ in our implementation so that the accepting probability p_n in the end is about $2^{-1/T_n} = 2^{-1/c} = 0.1$.

We now specify the score function of our RNA folding program. According to the Tinoco model [32], an RNA structure can be recursively decomposed into loops with independent free energy; the free energy of each loop is an affine function in the number of unpaired bases and the number of interior base pairs. Stacking pairs are the only type of loops without unpaired bases; therefore it has a negative free energy and stabilizes the RNA structure. Our scoring function is designed to approximate the number of stacking pairs [16,22,18].

For each *pair* of bases i and j adjacent in the lattice and at least a distance of 5 apart in the sequence (that is, $|i - j| \geq 5$), we assign it a *pair score* $s(i, j)$: 8 for a G-C, 5 for an A-U, 3 for a G-U, and -2 for the other pairs. Given a configuration, we compute for each base i the sum of absolute pair scores

$$sum(i) = \sum_{(i,j) \text{ a pair}} |s(i, j)|.$$

The *normalized score* of a pair (i, j) is then defined as

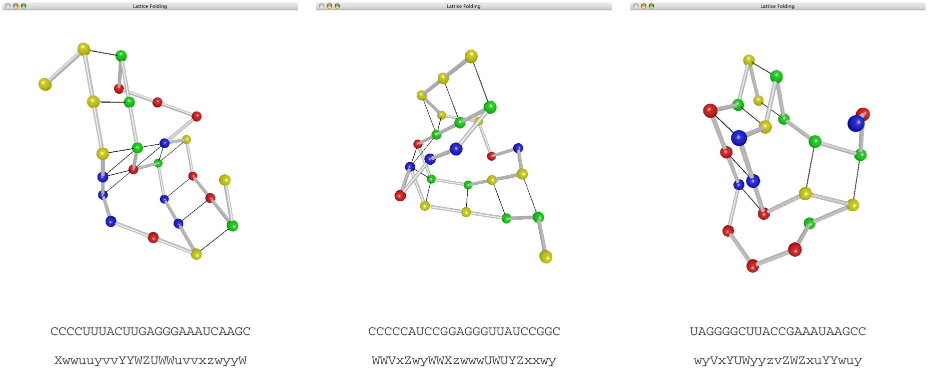
$$s'(i, j) = \frac{s(i, j)}{sum(i)} \cdot \frac{s(i, j)}{sum(j)} \cdot s(i, j),$$

where the two ratios $\frac{s(i,j)}{sum(i)}$ and $\frac{s(i,j)}{sum(j)}$, respectively, represent the levels of “commitment” of i and j in forming the pair (i, j). For a *stacking pair* SP that consists of two pairs (i, j) and $(i + 1, j - 1)$ such that $s'(i, j) > 0$ and $s'(i + 1, j - 1) > 0$, we define a *stacking score* $s''(SP) = s'(i, j) + s'(i + 1, j - 1)$. The score of a configuration is the total score of its stacking pairs.

By allowing each base to participate in multiple base pairs, our normalized scoring scheme avoids the expensive computation of a maximum weight matching [31] in the adjacency graph of each configuration; rare but still possible structures such as base triples [31] are not arbitrarily excluded by our model. We note especially that a pair (i, j) with a negative pair score -2 must also have a negative normalized score: the pair does not participate in stacking pairs, and its contributions to $sum(i)$ and $sum(j)$ only drag down the positive normalized scores of other pairs incident to either i or j. This implicitly discourages the unnecessary clustering of bases. Finally, we observe that the score change due to a pull move depends only on the relocated bases and their nearby bases. Since the average number of relocated bases of a random pull move is a small constant, we optimize the computation of the score change so that each simulated annealing step takes only constant time on average.

3 Experiment

To evaluate our RNA folding program, we used RNA sequences with known secondary structures from the PseudoBase [5]. The PseudoBase contained (as of February 1, 2007) 150 short RNA sequences with length between 20 and 40, which we used in our experiment. For each sequence of length ℓ, the `fold` options were set to `-z` $10^4\ell$ `-a` $10^4\ell$ $10^3\ell$ k. As k, the number of rounds, changes from 5 to 10 then to 20, the prediction accuracy (the fraction of base pairs correctly predicted) steadily increases from 53.7% to 65.7% then to 70.3%. For a sequence of length 40, a 20-round run of our RNA folding program typically takes about two minutes on an Apple iMac computer[2]. The prediction accuracy and the running time of our program compare favorably with those of the existing softwares [28,9] for predicting RNA secondary structures with pseudoknots.



CCCCUUUACUUGAGGGAAAUCAAGC

XwwuuyvvYYWZUWWuvvxzwyyW

(a) BSBV1

CCCCCAUCCGGAGGGUUAUCCGGC

WWVxZwyWWXzwwwUWUYZxxwy

(b) SBWMV2

UAGGGGCUUACCGAAAUAAGCC

wyVxYUWyyzvZWZxuYYwuy

(c) CcTMW_UPD-PK2

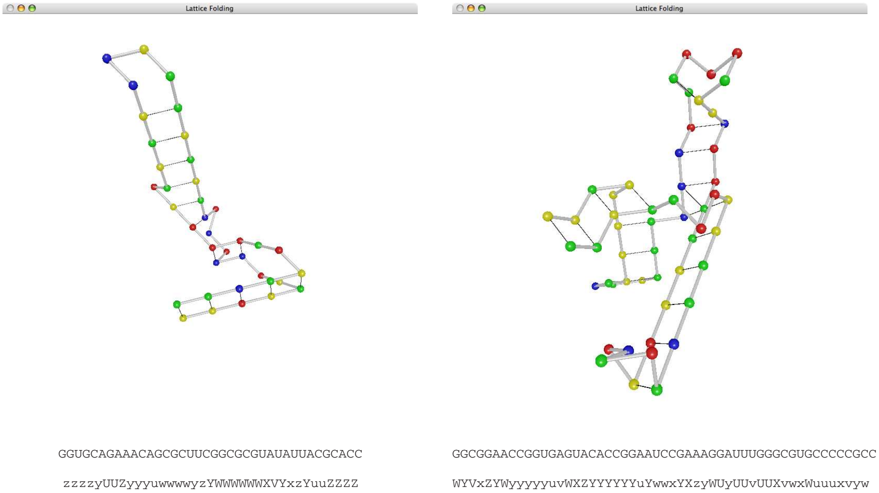
Fig. 5. Predicted structures of RNA sequences from the PseudoBase

In Fig. 5 and Fig. 6, we have sampled a few typical predictions made by our program: for each RNA sequence whose identifier is listed in the caption, the corresponding figure includes a screenshot of the visualizer output, the base sequence, and the predicted turn sequence. The four bases, A, C, G, U, are rendered in red, yellow, green, and blue, respectively; each base pair is rendered as a black line between the two bases. The three sequences in Fig. 5 (from the PseudoBase[3]) contain pseudoknots; the two sequences in Fig. 6 (from the Nucleic Acids Database[4]) do not contain pseudoknots. We note that the three sequences in Fig. 5 are also included as samples in Deogun et al.'s empirical study [9] of dynamic programming algorithms for RNA secondary structure prediction. Deogun et al.'s program correctly predicted the pseudoknots in BSBV1 and SBWMV2, but failed to identify the pseudoknot in CcTMW_UPD-PK2; our program correctly predicted the pseudoknots in all three sequences.

[2] 2GHz PowerPC G5 processor; 2GByte DDR SDRAM memory; MacOS 10.4.8; GCC 4.0.0.

[3] `http://wwwbio.leidenuniv.nl/~batenburg/pkb.html`

[4] `http://ndbserver.rutgers.edu/`



```
GGUGCAGAAACAGCGCUUCGGCGCGUAUAUUACGCACC

zzzzyUUZyyyuwwwwyzYWWWWWWXVYxzYuuZZZZ
```

(a) PDB:1B36

```
GGCGGAACCGGUGAGUACACCGGAAUCCGAAAGGAUUUGGGCGUGCCCCCGCC

WYVxZYWyyyyyuvWXZYYYYYYuYwwxYXzyWUyUUvUUXvwxWuuuxvyw
```

(b) NDB:UR0018

Fig. 6. Predicted structures of RNA sequences from the Nucleic Acids Database

With a relatively simple scoring scheme, our RNA folding program has already achieved very good prediction accuracy (around 70%) for short RNA sequences (Eddy [12] noted that "current RNA folding programs get about 50–70% of base pairs correct, on average"). In particular, as we can see directly from Fig. 5 and Fig. 6, simulated annealing with pull moves on a 3D triangular lattice can indeed predict correct RNA secondary structures with stacked base pairs. We see great potential for improving the prediction accuracy further when more elaborate energy parameters are incorporated in our program. Furthermore, our local search technique allows an easy trade-off between running time and prediction accuracy; more simulation steps usually results in better prediction accuracy, as our experimental result shows.

4 Concluding Remarks

In this paper, we introduced the Delta toolset for the structural analysis of biological sequences on a 3D triangular lattice. Our experiment with the proof-of-concept RNA folding program shows that it is both fast and accurate in predicting the secondary structures with pseudoknots of short RNA sequences.

We believe the lattice approach to the structural prediction of biological sequences is promising. There are several interesting directions for our future work. Beside experimenting with other effective move sets for the structural manipulation and exploring alternative optimization techniques such as tabu search and genetic algorithm, we intend to extend the Delta toolset for the manipulation and visualization of *multiple* sequences. This new feature will allow bioinformaticians to perform structural analysis of complex biological structures that contain protein-protein, protein-RNA, and RNA-RNA interactions.

References

1. J. P. Abrahams, M. van den Berg, E. van Batenburg, and C. Pleij. Prediction of RNA secondary structure, including pseudoknotting, by computer simulation. *Nucleic Acids Research*, 18(10):3035–3044, 1990.
2. R. Agarwala, S. Batzoglou, V. Dančík, S. E. Decatur, M. Farach, S. Hannenhalli, and S. Skiena. Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. In *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 390–399, 1997.
3. T. Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1-3):45–62, 2000.
4. F. H. D. van Batenburg, A. P. Gultyaev, and C. W. A. Pleij. An APL-programmed genetic algorithm for the prediction of RNA secondary structure. *Journal of Theoretical Biology*, 174(3):269–280, 1995.
5. F. H. D. van Batenburg, A. P. Gultyaev, C. W. A. Pleij, J. Ng, and J. Oliehoek. Pseudobase: a database with RNA pseudoknots. *Nucleic Acids Research*, 28(1):201–204, 2000.
6. B. Berger and T. Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. In *Proceedings of the 2nd Conference on Computational Molecular Biology (RECOMB'98)*, pages 30–39, 1998.
7. P. Crescenzi, D. Goldman, C. Papadimitriou, A. Piccolboni, and M. Yannakakis. On the complexity of protein folding. In *Proceedings of the 2nd Conference on Computational Molecular Biology (RECOMB'98)*, pages 61–62, and in *Proceedings of the 30th Annual ACM Symposium on Theory of Computing (STOC'98)*, pages 597–603, 1998.
8. M. Crochemore, D. Hermelin, G. M. Landau, and S. Vialette. Approximating the 2-interval pattern problem. In *Proceedings of the 13th Annual European Symposium on Algorithms (ESA'05)*, LNCS 3669, pages 426–437, 2005.
9. J. S. Deogun, R. Donis, O. Komina, and F. Ma. RNA secondary structure prediction with simple pseudoknots. In *Proceedings of the 2nd Asia-Pacific Bioinformatics Conference (APBC'04)*, pages 239–246, 2004.
10. K. A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.
11. K. A. Dill. Dominant forces in protein folding. *Biochemistry*, 29:7133–7155, 1990.
12. S. R. Eddy. How do RNA folding algorithms work? *Nature Biotechnology*, 22(11):1457–1458, 2004.
13. A. P. Gultyaev, F. H. D. van Batenburg, and C. W. A. Pleij. The computer simulation of RNA folding pathways using a genetic algorithm. *Journal of Molecular Biology*, 250(1):37–51, 1995.
14. W. E. Hart and S. Istrail. Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal. In *Proc. 27th Annual ACM Symposium on Theory of Computing (STOC'95)*, pages 157–168, 1995.
15. I. L. Hofacker, W. Fontana, P. F. Stadler, S. Bonhoeffer, M. Tacker, and P. Schuster. Fast folding and comparison of RNA secondary structures. *Monatshefte für Chemie*, 125(2): 167–188, 1994.
16. S. Ieong, M.-Y. Kao, T.-W. Lam, W.-K. Sung, and S.-M. Yiu. Predicting RNA secondary structure with arbitrary pseudoknots by maximizing the number of stacking pairs. *Journal of Computational Biology*, 10(6):981–995, 2003.
17. M. Jiang. A 2-approximation for the preceding-and-crossing structured 2-interval pattern problem. *Journal of Combinatorial Optimization*, Special Issue on Bioinformatics, to appear.
18. M. Jiang. Improved approximation algorithms for predicting RNA secondary structures with arbitrary pseudoknots. In *Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management (AAIM'07)*, to appear.

19. M. Jiang and B. Zhu. Protein folding on the hexagonal lattice in the HP model. *Journal of Bioinformatics and Computational Biology*, 3(1):19–34, 2005.
20. S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
21. N. Lesh, M. Mitzenmacher, and S. Whiteslides. A complete and effective move set for simplified protein folding. In *Proceedings of the 7th Annual International Conference on Computational Molecular Biology (RECOMB'03)*, pages 188–195, 2003.
22. R. B. Lyngsø. Complexity of pseudoknot prediction in simple models. In *Proceedings of the 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, pages 919–931, 2004.
23. R. B. Lyngsø and C. N. S. Pedersen. RNA pseudoknot prediction in energy-based models. *Journal of Computational Biology*, 7(3/4):409–427, 2000.
24. R. B. Lyngsø, M. Zuker, and C. N. S. Pedersen. Fast evaluation of interval loops in RNA secondary structure prediction. *Bioinformatics*, 15(6):440–445, 1999.
25. G. Mauri, G. Pavesi, and A. Piccolboni. Approximation algorithms for protein folding prediction. In *Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'99)*, pages 945–946, 1999.
26. A. Newman. A new algorithm for protein folding in the HP model. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'02)*, pages 876–884, 2002.
27. R. Nussinov, G. Pieczenik, J. R. Griggs, and D. J. Kleitman. Algorithms for loop matching. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 1978.
28. E. Rivas and S. R. Eddy. A dynamic programming algorithm for RNA structure prediction including pseudoknots. *Journal of Molecular Biology*, 285:2053–2068, 1999.
29. J. Ruan, G. D. Stormo, and W. Zhang. An iterated loop matching approach to the prediction of RNA secondary structure with pseudoknots. *Bioinformatics*, 20(1):58–66, 2004.
30. B. A. Shapiro and J. C. Wu. Predicting RNA H-type pseudoknots with the massively parallel genetic algorithm *Computer Applications in the Biosciences*, 13(4):459–471, 1997.
31. J. E. Tabaska, R. B. Cary, H. N. Gabow, and G. D. Stormo. An RNA folding method capable of identifying pseudoknots and base triples. *Bioinformatics*, 14(8):691–699, 1998.
32. I. Tinoco, P. N. Borer, B. Dengler, M. D. Levine, O. C. Uhlenbeck, D. M. Crothers, and J. Gralla. Improved estimation of secondary structure in ribonucleic acids. *Nature New Biology*, 246:40–42, 1973.
33. Y. Uemura, A. Hasegawa, S. Kobayashi, and T. Yokomori. Tree adjoining grammars for RNA structure prediction. *Theoretical Computer Science*, 210(2):277–303, 1999.
34. S. Vialette. On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science*, 312:223–249, 2004.
35. M. S. Waterman. *Introduction to Computational Biology: Maps, Sequences and Genomes.* Chapman & Hall, 1995.
36. M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Research*, 9(1):133–148, 1981.

Statistical Estimate for the Size of the Protein Structural Vocabulary

Xuezheng Fu[1], Bernard Chen[1], Yi Pan[1], and Robert W. Harrison[1,2]

[1] Department of Computer Science, Georgia State University, Atlanta, GA, 30303
[2] Department of Biology, Georgia State University, Atlanta, GA, 30303
xfu1@student.gsu.edu, bchen3@cs.gsu.edu, pan@cs.gsu.edu, rharrison@cs.gsu.edu

Abstract. The concept of structural clusters defining the vocabulary of protein structure is one of the central concepts in the modern theory of protein folding. Typically clusters are found by a variation of the K-means or K-NN algorithm. In this paper we study approaches to estimating the number of clusters in data. The optimal number of clusters is believed to result in a reliable clustering. Stability with respect to bootstrap sampling was adapted as the cluster validation measure for estimating the reliable clustering. In order to test this algorithm, six random subsets were drawn from the unique chains in the PDB. The algorithm converged in each case to unique set of reliable clusters. Since these clusters were drawn randomly from the total current set of chains, counting the number of coincidences and using basic sampling theory provides a rigorous statistical estimate of the number of unique clusters in the dataset.

Keywords: K-means algorithm, clustering, stability validation measure.

1 Introduction

Understanding the protein structure motifs and sequence-structure correspondence is an important task in current bioinformatics research. In this work, we explored these problems using the K-means algorithm. K-means algorithm is by far the most popular clustering method used in scientific and industrial applications [1]. However, its robustness is heavily affected by the initial number of clusters K, and in general, there is no reliable algorithm for predicting K. A variety of methods have been proposed on how to initialize the K or on how to let the clustering depend less on the initial K. All these efforts try to discover the optimal number of clusters for a dataset. Since stability has been widely used as a validation measure and proved to have good performance [2-4], in this work we decided to investigate the stability-based measure formulated by Ben-hur et al. [2]. for reliable clustering of protein structures. We proposed a new descriptor for representation of the protein structures and compared it with the descriptor used by Chen et al. [5].

In this work six random subsets drawn from the current set of chains and used as test cases to show the algorithm converged. Since these are random subsets, the probability of coincidences between each subset allows the estimation of the total number of clusters that could be found on the whole data set. The estimates we retrieve

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 530–538, 2007.

are consistent with the direct observations of Zhong et al. [6]. Therefore population sampling coupled with reliable clustering is a viable approach to estimating K for the total population.

2 Method

The meaning behind of the stability is that a meaningful cluster should appear on all bootstrap samplings of the data. In bootstrap sampling data are drawn from the dataset with replacement thus preserving the underlying distribution. Based on this idea, it is not necessary to make any assumption about the distribution of the data or the shape of the clusters. This is the most significant advantage of this algorithm. The algorithm of determining the reliable clustering is outlined in Table 1.

Table 1. Algorithm of detecting reliable clustering

```
Input: Kmax{user-defined maximum number of clusters},
       Smax{user-defined maximum number of resample}
Output: R{cluster is reliable at R}
1. Produce the mean similarity of each k
    For k=2 to Kmax do
       Resample Smax new datasets from the original one;
       Apply K-means clustering on each dataset;
       Compute cluster similarity of any two datasets;
       Compute the mean of the similarities;
    End For
2. The largest mean similarity occurs at $k_1$;
3. The second largest mean similarity occurs at $k_2$;
4. Return the reliable R
     If isReliable($k_1$) is true, R=$k_1$
     Elseif isReliable($k_2$) is true, R=$k_2$
     Else R=0
     Return R
```

In Table 1, `isReliable(k)` is a function to judge whether the clustering at k is reliable. Given the k, if the distribution of mean similarities before k is increasing while the distribution after k is decreasing, `isReliable(k)` is true. If the return value is 0, it indicates that there is no structure in the dataset. The cluster similarity mentioned in the Table 1 is defined in section 2.2.

2.1 Cluster Similarity

Cluster similarity is the estimation of similarity between any two clustering solutions on the same dataset. It is given by

Let X is the dataset to be clustered, $X=\{x_1, x_2,\ldots, x_N\}$ where x_i is R^d ,N is the number of samples in the dataset. X is clustered into k clusters. We use a NxN matrix

C to indicate whether any two samples in X are in the same cluster. If x_i and x_j are in the same cluster and i is not equal to j, C_{ij} is 1. Otherwise C_{ij} is 0.

For two datasets P and Q, their matrix representations of clustering solutions are respectively C^P and C^Q. The number of common joined pairs in clustering solutions of P and Q can be described the following dot product

$$r = < C^P, C^Q > = \sum_{i,j} C^P_{i,j} C^Q_{i,j} \quad (1)$$

The cluster similarity proposed by Fowlkes et al. [7] is then defined as

$$S(P,Q) = \frac{r}{\sqrt{|C^P| \times |C^Q|}} \quad (2)$$

where $| C^P | = < C^P, C^P >$.

According to Cauchy-Schwartz inequality: $< C^P, C^Q > \leq \sqrt{< C^P, C^P > \times < C^Q, C^Q >}$, the similarity is between 0 and 1. 1 represents that two clustering solutions are identical. Equation (2) is the cluster similarity used in this experiment.

2.2 Initialization of K-Means

The initialization of K-means is to set k cluster centroids for a dataset. It has been reported that results obtained from the K-means are dependent on the initialization of the cluster centroids [8].

In this work, we initialize the K-means using an algorithm which tries to isolate the initial centroids as far as possible. It includes four steps to find the K initial centroids. First, randomly choose a sample of the dataset as the first initial centroid. Second, search for the second sample which is farthest from the first centroid. Third, search for the next sample which his farthest from all previous centroids. Firth, repeat steps 3 till K centroids are found.

3 Empirical Analyses

3.1 Dataset

Six datasets are used in this experiment and each dataset includes 100 protein structures. The protein structures are randomly selected from Protein Sequence Culling Server [9]. In [5], the protein sequences are separated into segments using sliding window with a window size 9. To compare with their work, we keep using 9 as the window size. The protein structures in the six datasets are represented in two descriptors. Our descriptor generates structure segments while the descriptor [5] produces sequence segments. The definitions of structure segments and sequence segments are explained as follows.

Representation of structure segments

A good descriptor should contain enough information and keep as easy-understanding and each-using as possible for study. Keeping this goal in mind, we propose a new descriptor for the protein structure segments. The descriptor is based on a distance matrix which consists of the Euclidean distance between any two amino acids in a sliding window. The definition of distance matrix is described as follows:

Let the sliding window size is w. The subsequence in the sliding window is S=X1X2…Xw. The coordinates of a protein backbone are known. The distance matrix (DM) is

$$DM = \begin{cases} d_{ij}, & if\ i < j <= w \\ 0, & if\ i >= j \end{cases} \tag{3}$$

where d_{ij} is the Euclidean distance between amino acids i and j; w is the sliding window size.

The DM can be further simplified into a vector which only contains d_{ij} where i<j<=w. Consequently, a structure segment is finally described as a vector

$$\mathrm{V} = \{\mathrm{d}_{ij} \mid 0 < \mathrm{i} < \mathrm{j} <= \mathrm{w}\} \tag{4}$$

Representation of sequence segments

We use the frequency profile from the HSSP [10] to represent the sequence segments. The frequency profile from HSSP is constructed based on the alignment of each protein sequence from the homologous sequences in protein data bank [11]. The secondary structures of each protein sequence are obtained from DSSP [12], a database of secondary structure assignments for all protein entries in the PDB. DSSP originally assigns the secondary structure to eight different classes. In this work, we combine the eight classes into three classes: H, G and I to H (Helices); B and E to E (Sheets); all others to C (Coils).

3.2 Parameters Setup

In this experiment, we set the Kmax (the maximum number of clusters) change from 2 to 50 and Smax (the maximum number of sampling) as 20.

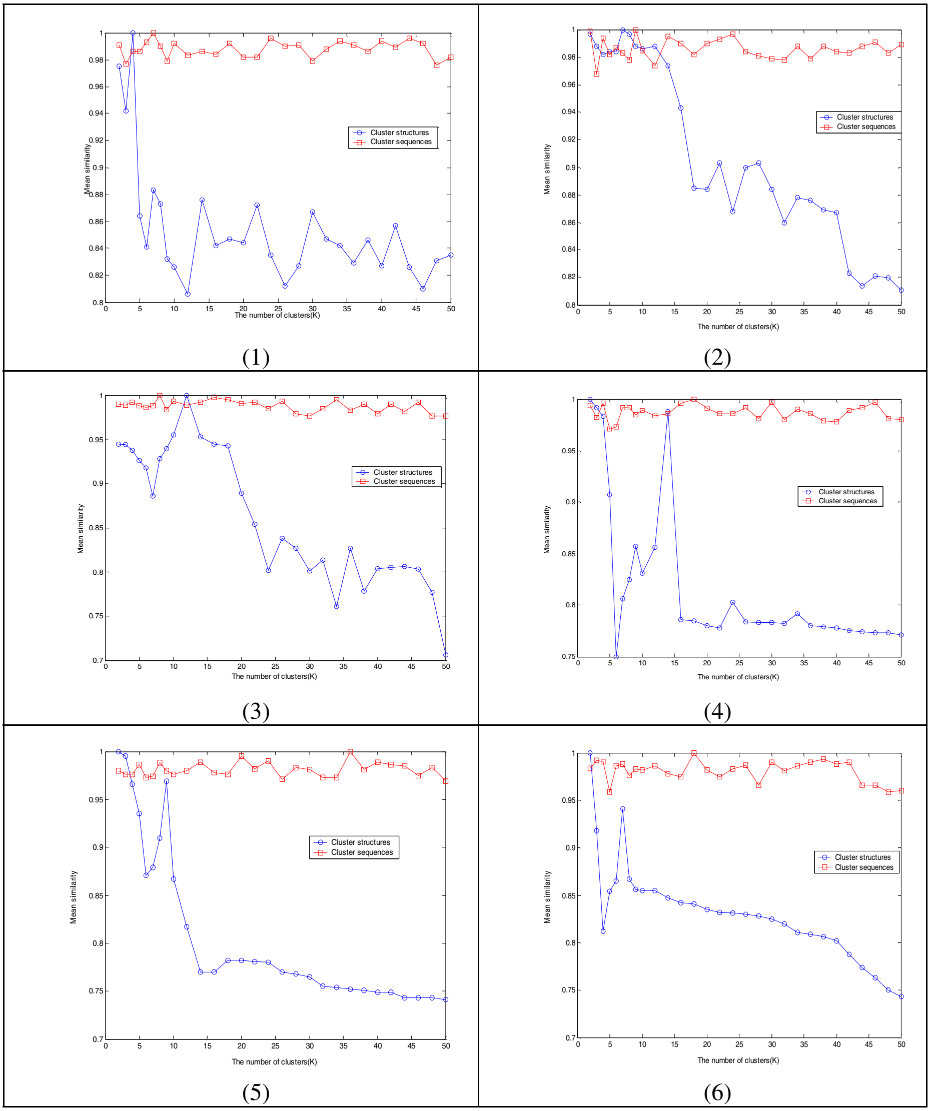
3.3 Experimental Results

The distribution of mean similarities of each protein structure dataset is shown in Fig.1.

Fig. 1(1)-(6) are respectively the distributions of six protein datasets, including both clustering structure segments and clustering sequence segments. We observe the following advantages of structure segments over sequence segments:

- Compared with the mean similarity values using structure segments, the mean similarity values scatter in a small range using sequence segments. For the six datasets, the mean similarity values using sequence segments respectively range in [0.976, 1], [0.969, 1], [0.977, 1], [0.971, 1], [0.973, 1], [0.959, 1]. Using structure segments, the mean similarity values using structure segments respectively range in [0.806, 1], [0.811, 1], [0.761, 1], [0.750, 1], [0.741, 1], [0.743, 1]. Larger range of mean similarity values favors the determination of optimal number of clusters.

- Unique maximum mean similarity value is found in every dataset no matter using sequence segments or structure segments.
- It is more stable to find the optimal number of clusters using structure segments than using sequence segments. In our experiment, clustering on structure segments works well for all datasets while clustering on sequence segments works for none of the six datasets.
- Using structure segments needs fewer dimensions than using sequence segments. For window size 9 in this experiment, the former is 36 and the later is 180.



(1)

(2)

(3)

(4)

(5)

(6)

Fig. 1. Distribution of mean similarities. (1)-(6) are respectively the distributions of six datasets.

3.4 Evaluation of Experimental Results

3.4.1 Secondary Structure Similarity Analysis

To verify the effectiveness of our clustering analysis, a biological evaluation criterion is used to evaluate the optimal number of clusters we found. The optimal number of clusters for each protein dataset has been found for the clustering of structure segments. For the clustering of sequence segments, the K where the maximum mean similarity occurs is adapted as the optimal number of clusters. The biological evaluation criterion is called structural secondary structure similarity [10] which is used in many literatures. Its definition is given by,

For each cluster, the secondary structure similarity of a cluster is

$$\text{Secondary Structure Similarity} = \frac{\sum_{i=1}^{ws} \max(p_{i,H}, p_{i,E}, p_{i,C})}{ws} \tag{5}$$

Where ws is the window size and $p_{i,H}$ shows the frequency of occurrence of helix among the segments of a cluster at position i. $p_{i,E}$ and $p_{i,C}$ are defined in a similar way.

Sander and Schneider state [10], if the structural secondary structure similarity for a cluster is larger than 70%, the cluster can be considered structurally identical. If the secondary structure similarity of a cluster exceeds 60% and bellows 70%, the cluster can be considered weakly structurally homologous [6].

According to the secondary structure similarity definition above, we calculate the secondary structure similarity of all segments of each dataset. It is shown in Table 2.

Table 2. Secondary structure similarity of each protein dataset

Datasets	**1**	**2**	**3**	**4**	**5**	**6**
# of segments (window size =9)	18797	16138	17866	16365	17454	16490
SSS of all segments	0.392	0.420	0.389	0.397	0.384	0.388

SSS: Secondary Structure Similarity

We do experiments by clustering both structure segments and sequence segments. The average secondary structure similarity analysis is shown in the Table 3.

Table 3. The average secondary structure similarity of each protein dataset

Datasets		**1**	**2**	**3**	**4**	**5**	**6**
Structure segments	Optimum K	4	7	12	14	9	7
	Arg. SSS	0.616	0.622	0.631	0.615	0.623	0.624
Sequence segments	Optimum K	7	9	8	18	20	20
	Arg. SSS	0.410	0.420	0.398	0.411	0.401	0.407

Arg. SSS: Average Secondary Structure Similarity

To obtain the average secondary structure similarity for each dataset, we first cluster the dataset using the optimal number of clusters found in Figure 1. Then we compute the secondary structure similarity of each cluster and the average secondary structure similarity. From the table 3, we can see that the average secondary structure similarity of each dataset by clustering structure segments is above 0.6. However, clustering sequence segments gives an average secondary structure similarity about 0.4 which is much lower than the method of clustering structures. According to [5], the clusters we found have meaningful secondary structural similarity. It proves that the distance descriptor of structure segments is effective and our algorithm is helpful to find the optimal number of clusters and work well with the distance descriptor for protein structure analysis.

3.4.2 Population Analysis

The total population can be estimated by extracting and tagging a subset of a population, releasing the tagged subset and then estimating the probability of recovering the tagged individual when a new sample is drawn from the population. Since the probability of recovering a tagged individual is:

$$P(Tagged\ Individual) = \frac{N_{tagged}}{N_{total}} \approx \frac{N_{tagged\,in\,sample}}{N_{sample}} \tag{6}$$

Where N_{tagged} is the number of tagged individuals in the total population and the N_{total} is the number of total population; N_{sample} is the number of individuals in a sample and $N_{tagged\ in\ sample}$ is the number of tagged individuals in a sample. That is, $N_{tagged\ in\ sample}$ is the number of coincidence between N_{tagged} and N_{sample}. From the above equation, the N_{total} can be derived as:

$$N_{total} = \frac{N_{tagged}}{P(Tagged\ \ Indivudal\ )} \approx \frac{N_{tagged}\,N_{sample}}{N_{tagged\ \ in\ sample}} \tag{7}$$

Thus, individual estimates for the total population size can be made by counting the number of coincidences between clusters. In our experiment, we have found the number of clusters for each of the six protein datasets. Based on the knowledge, we can estimate the range of how many clusters are in a larger or even the whole set of protein structures. The number of clusters in two different datasets can be seen as N_{tagged} and N_{sample} respectively. The number of coincidences between clusters of two different datasets is $N_{tagged\ in\ sample}$.

Finding the mean and standard deviation of these individual estimates determines a statistical confidence interval. Table 4 shows the number of coincidences between different subsets. The upper triangle in Table 4 shows when clusters were defined to be identical when the RMSD between them was less than 1Å; the lower triangle shows the number of coincidences with a criterion of less than 2Å. The diagonal in Table 4 shows the number of clusters in each dataset.

Not surprisingly, there are more coincidences with the looser criteria (less than 2Å), and therefore there will be a lower total number when the looser criterion is used.

Table 4. The number of coincidences between clusters

Datasets	1	2	3	4	5	6
1	4	0	0	0	0	0
2	0	7	2	1	1	5
3	0	2	12	3	2	1
4	0	2	5	14	1	2
5	1	3	3	3	9	2
6	0	7	2	3	4	7

According to equation 7, we now calculate the N_{total} for each pair of different datasets. Let's take the dataset 2 and 3 as an example. N_{tagged} is 7. N_{sample} is 12. $N_{tagged\ in\ sample}$ is 2 in the upper triangle of Table 4. Then N_{total}=7*12/2=42. Computing in this way, we get the total estimate from each pair of datasets. With the 1Å criterion the mean estimate for the total number of 9-mers was 62 with the standard deviation of 30.8. Thus the three-sigma estimate for the maximum number of 9-mers is 154. When the larger 2Å criterion is used the mean value is 32.5 with a standard deviation of 12.7. Thus the three-sigma estimate for the maximum number of 9-mers is 70.

For comparison, Zhong et al [6] found 211-253 clusters of 9-mers with >60% secondary structure similarity and 80-92 clusters with >70% secondary structure similarity when using different K-means algorithms. Thus our statistical estimates are consistent with direct observation.

4 Conclusion

In this paper, we proposed a new descriptor for representation of protein structures and an algorithm for detecting reliable clustering. In order to test our algorithm, six random datasets of protein three dimensional structures are drawn from the unique chains in the PDB. The algorithm converges in each case to a unique set of reliable clusters. Since these clusters are drawn randomly from the total current set of chains, counting the number of coincidences and using basic sampling theory provides a rigorous statistical estimate of the number of unique clusters in the dataset. The effectiveness of our algorithm with the new descriptor was verified by the secondary structure similarity analysis. The population estimates derived from sampling were consistent with the direct observation.

Acknowledgements

This work was partially supported by NIH1 P20 GM065762-01A1, the Georgia Research Alliance and the Georgia Cancer Collation. Dr. Robert Harrison is a Georgia Cancer Collation Distinguished Scholar. Xuezheng Fu is supported by Molecular Basis for Disease (MBD) Doctoral Fellowship.

References

1. A. K. Jain, M. N. Murty, P. J. Flynn Data clustering: a review. ACM Computing Surveys, Volume 31, Issue 3, (1999) pp.264-323.
2. A. Ben-Hur, A. Elisseeff and I. Guyon. A stability based method for discovering structure in clustered data. Pacific Symposium on Biocomputing, (2002) pp.6-17.
3. J. Bryan, Problems in gene clustering based on gene expression data. Journal of Multivariate Analyis. 90, (2004) pp.67-89.
4. S. Dudoit and J. Fridlyand, A prediction-based resampling method to estimate the number of clusters in a dataset. Genome Biology 3, (2002) pp.0036.1-0036.21.
5. B. Chen, P.C. Tai, R. Harrison, and Y. Pan, FIK model: A Novel Efficient Granular Computing Model for Protein Sequence Motifs and Structure Information Discovery, IEEE BIBE 2006 proceeding, (2006) pp. 20-26.
6. W. Zhong, G. Altun, R. Harrison, P.C. Tai and Y. Pan, Improved K-Means Clustering algorithm for Exploring Local Protein Sequence motifs Representing Common Structural Property, IEEE transactions on Nanobioscience, vol4, no.3, (2005) pp. 255-265.
7. E.B. Fowlkes and C.L. Mallows, A method for comparing two hierarchical clusterings. Journal of the American Statistical Association 78(383), (1983) pp. 553–584.
8. J. Pena, J. Lozano and P. Larranaga, An Empirical comparison of four initialization methods for the k-means algorithm, Pattern Recognition Letters Vol. 20, (1999) pp. 1027-1040.
9. G. Wang and R. L. Dunbrack, Jr., PISCES: a protein sequence-culling server, Bioinformatics, vol. 19, no. 12, (2003) pp.1589-1591.
10. C. Sander and R. Schneider, Database of similarity derived protein structures and the structure meaning of sequence alignment, Proteins:Struct. Funct. Genet., vol.9 no. 1, (1991) pp. 56-68.
11. H.M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T.N. Bhat, H. Weissig, I.N. Shindyalov, and P.E Bourne, The Protein Data Bank. Nucleic Acids Res., (2000) 28, pp. 235–242.
12. W. Kabsch and C. Sander, Dictionary of protein secondary structure: Pattern recognition of hydrogen-bonded and geometrical features, Biopolymers, vol. 22, (1983) pp. 2577–2637.

Coclustering Based Parcellation of Human Brain Cortex Using Diffusion Tensor MRI

Cui Lin[1], Shiyong Lu[1], Danqing Wu[1], Jing Hua[1], and Otto Muzik[2]

[1] Department of Computer Science, Wayne State University
{cuilin,shiyong,dqwu,jinghua}@wayne.edu
[2] PET Center at Children's Hospital of Michigan, Radiology at Wayne State University
otto@pet.wayne.edu

Abstract. The fundamental goal of computational neuroscience is to discover anatomical features that reflect the functional organization of the brain. Investigations of the physical connections between neuronal structures and measurements of brain activity in vivo have given rise to the concepts of anatomical and functional connectivity, which have been useful for our understanding of brain mechanisms and their plasticity. However, at present there is no generally accepted computational framework for the quantitative assessment of cortical connectivity. In this paper, we present accurate analytical and modeling tools that can reveal anatomical connectivity pattern and facilitate the interpretation of high-level knowledge regarding brain functions are strongly demanded. We also present a coclustering algorithm, called Business model based Coclustering Algorithm (BCA), which allows an automated and reproducible assessment of the connectivity pattern between different cortical areas based on Diffusion Tensor Imaging (DTI) data. The proposed BCA algorithm not only partitions the cortical mantel into well-defined clusters, but at the same time maximizes the connection strength between these clusters. Moreover, the BCA algorithm is computationally robust and allows both outlier detection as well as operator-independent determination of the number of clusters. We applied the BCA algorithm to human DTI datasets and show good performance in detecting anatomical connectivity patterns in the human brain.

1 Introduction

With ever-improving imaging technologies, the complexity and scale of brain imaging data has continued to grow at an explosive pace. Recent advances in imaging technologies, especially that of Diffusion Tensor Imaging [1,2,3], have allowed an increased understanding of normal and abnormal brain structure and function [4,3]. It is well understood that normal brain function is dependent on the interactions between specialized functional areas of the brain which process information within local and global networks. Perhaps the most promising approach to parcelate the cerebral cortex into such distinct functional areas originates from the notion that functionally discrete areas of the cortical mantel are characterized by cortico-cortical connectivity patterns, which represent functionally integrated neural subsystems and determine the region's functional properties [5] and also allow their anatomical delineation and mapping.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 539–550, 2007.

At present, no generally accepted parcellation scheme exists for the human cortex, although circumstantial evidence points to a distinct arrangement of functional territories within the cortex. As illustrated in Figure 1, most cortical voxels in one region are strongly connected to a particular region of the cortex and the connections to any other regions are relatively weaker. For example, most voxels on the top of the cortex congregating in cortical region C_2 are connected to voxels of cortical region C_1, with only few connections to other cortical regions. Therefore, in order to perform an accurate in-vivo analysis of the cortico-cortical connectivity, what is needed is a partitioning procedure that not only simultaneously partitions voxels into groups, but also identifies the corresponding strong connectivities between the two classes of groups.

Traditional clustering algorithms [6,7,8,9] are suboptimal in incorporating anatomical constraints and as a result will fail to identify accurately the corresponding connectivity between cortical regions. Moreover, our focus in this paper is to assess the neural connections within a hemisphere (intra-hemispheric connections) sine these connections are relatively weak compared to the connections between the left and right hemisphere, but represent crucial neural pathways which are abnormal in neurological disease. Consequently, we consider only clusters which connect cortical areas within one hemisphere.

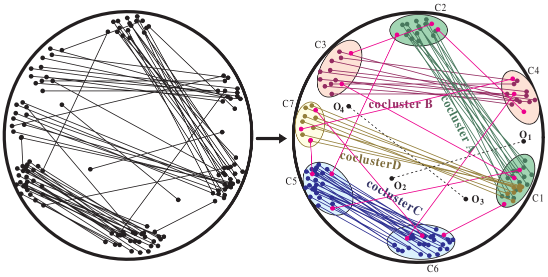


Fig. 1. The coclustering process

The main contributions of this paper are:

1. We are the first to propose a new coclustering model for defining cortico-cortical connectivity analysis as a computational problem.
2. Our BCA coclustering algorithm is able to define functional cortical areas based on cortico-cortical fiber tract connections taking into account anatomical constraints. In contrast to traditional clustering paradigms, the BCA algorithm is not only able to partition image voxels within the cortical mantel into well-defined clusters, but also is able to maximize the connectivity strength between such clusters. Moreover, the BCA method is able to identify outliers as well as the number of cortical clusters with high efficiency.
3. The application of the BCA algorithm to human DTI dataset allows automated and reproducible assessment of the connectivity patterns in the human brain.

Organization. The rest of the paper is organized as follows: Section 3 formalizes the coclustering model for the cortico-cortical connectivity analysis. Section 4 proposes our coclustering algorithm, BCA, in order to assess fiber tract connectivity between remote cortical areas. Section 5 presents 3-D visualization of the obtained results followed by a discussion of the application in patient groups. Finally, Section 6 concludes the paper and comments on future work.

2 Background and Related Work

The cerebral cortex sends connections (efferents) and receives connections (afferents) from many subcortical structures, but the largest part of the connections arriving at the cerebral cortex comes from the cerebral cortex itself. Assessing connectivity patterns of cortico-cortical fiber tracts is important for our understanding of the mechanisms involved in human brain functions and might provide clues towards the identification and characterization of many neurological diseases.

Recently, Diffusion Tensor Imaging (DTI) tractography has been shown to produce results that are consistent with known pathways formed by major white matter fiber tracts in the human brain [10,2], although limitations in data acquisition and processing algorithms [11] related to clinical constraints produce data which cannot resolve crossing or intersecting fibers. DTI is based upon the ability of MRI to evaluate in vivo the direction and magnitude of water diffusion in tissues [2]. These attributes of in vivo water diffusion depend upon microscopic tissue architecture [12]. Therefore, changes in these parameters serve as markers for changes in tissue micro-architecture. The principal eigenvector obtained from DTI provides information about the preferential direction of water diffusion in each imaging voxel. This direction corresponds to the direction of the nerve fiber bundles, which predominantly constitute the given voxel. Hence, different nerve fiber bundles can be identified and used to assess the integrity of white matter tracts throughout the brain [13,12]. Despite some success in delineating functional cortical areas using DTI, a systematic framework allowing functional parcellation of the neocortex based on quantitative assessment of fiber tract connectivity has not yet been produced, and the relationship among cortical territories, fiber tracts, and neuronal connections remains controversial. Consequently, there is a need to further develop advanced clustering algorithms that allow better characterization of brain connectivity patterns and as a result improve our understanding of process interactions in a complex biological system.

Traditional partitioning relocation clustering algorithms, such as the K-means [6], K-medoids [7] are simple and efficient, however, their final results may be overly sensitive to the initial cluster set and the presence of outliers. In addition, it is difficult to implement when no information exists about the likely cluster number. Hierarchical clustering algorithms [8,14] do not require the number of clusters K as input, but they require a termination condition. In addition, they do not support reclassification of objects to new clusters. Density-based algorithms [9,15,16] have good performance with respect to noise handling and one-scan efficiency, but are suboptimal for the cortico-cortical problem, as they do not consider the connectivity strength between clusters, hence fail to identify accurately the corresponding strongest connectivity between cortical regions.

Even though our first coclustering algorithm GCA [17] was effective in the analysis of thalamo-cortical connectivity, it is not directly applicable to the cortico-cortical connectivity problem as each fiber connects to two different cortical voxels. Direct application of the GCA algorithm to cortico-cortical connectivity analysis might lead to the following undesirable results: (1) the same voxel can be classified simultaneously into several clusters, (2) two end voxels of a fiber tract might be classified into the same cluster, and (3) two partitions of voxels given by GCA might be inconsistent and the resolution of this inconsistency is not obvious.

3 The Coclustering Model

In this section, we present our coclustering model, which models the cortico-cortical connectivity problem. The structure of the cerebral cortex and its cortical connections is initialized as a graph $G(V, F)$, as illustrated in Figure 1, where V represents all cortical voxels, and F represents all of the cortico-cortical connections between each other.

Definition 1 (Outlier)**.** *Given a partition C of $G(V, F)$, $C = \{C_1, C_2, \cdots, C_K\}$, an outlier is defined as:*

$$o = \{v | v \in V, \forall C_i \in C, v \notin C_i\}$$

Definition 2 (Connection strength $\theta(C_i, C_j)$)**.** *Given a cortical cluster C_i and C_j, the connection strength between C_i and C_j is defined as:*

$$\theta(C_i, C_j) = \frac{N_{ij}}{|C_i|}$$

where N_{ij} equals to the total number of connections between C_i and C_j.

Definition 3 (Spouse cluster)**.** *Given a partition P of $G(V, F)$, $P = \{C_1, C_2, ..., C_k\}$, $SP(C_i) = \{C_j | \forall C_k \in P, \theta(C_i, C_j) \geq \theta(C_i, C_k)\}$*

Definition 4 (Cocluster set)**.** *$< C_i, C_j >$ is called a* cocluster set *iff C_j is the spouse cluster of C_i.*

Example 1. In Figure 1, C_2 is C_1's spouse cluster, they form a cocluster set $cocluster A$ $< C_1, C_2 >$, $< C_7, C_1 >$ forms another cocluster set $cocluster D$. worth mentioning, the two elements in a cocluster set are not commutative. thus, $< C_i, C_j >$ does not necessarily implies $< C_j, C_i >$. For instance, $< C_1, C_7 >$ is an invalid cocluster set, because only C_2 can be identified as C_1's spouse cluster.

The goal of our coclustering procedures is to partition objects into groups while minimizing the cross-connectivity costs between those groups. More specifically, the coclustering procedures will separate objects into K groups so that (1) similar objects are within the same group, while dissimilar objects are in different groups, (2) there is a one-to-one correspondence /one-to-many correspondence between one cortical cluster to another / other clusters; and (3) the total cross-connectivity cost between each cluster and its non-spouse cluster is minimized.

To achieve the above goals, we define several notions. First, we define the centroid of a cluster and its Within-Cluster Variation (WCV) to quantify the similarity of objects within one cortical cluster.

Definition 5 (Centroid). *Given a cortical cluster C_k, its centroid $\overrightarrow{\mu_k}$ is defined as:*

$$\overrightarrow{\mu_k} = \frac{\sum_{\overrightarrow{X_n} \in C_k} \overrightarrow{X_n}}{|C_k|}$$

where $|C_k|$ represents the number of cortical voxels in cluster C_k.

Definition 6 (WCV). *We define Within-Cluster Variation of cortical cluster C_k as:*

$$WCV(C_k) = \sum_{\overrightarrow{X_n} \in C_k} d(\overrightarrow{X_n}, \overrightarrow{\mu_k})$$

where $d(\overrightarrow{X_n}, \overrightarrow{\mu_k})$ is the Euclidean distance between the cortical voxel $\overrightarrow{X_n}$ and the centroid $\overrightarrow{\mu_k}$ of cortical cluster C_k.

Second, we define the Total Within-Cluster Variation(*TWCV*) to quantify the quantity of a particular partitioning.

Definition 7 (TWCV). *The Total Within-Cluster Variation of a cortical partition (C_1,· · ·, C_K) is defined as*

$$\begin{aligned} & TWCV(C_1, \cdots, C_K) \\ &= \sum_{k=1}^{K} WCV(C_k) \\ &= \sum_{k=1}^{K} \sum_{\overrightarrow{X_n} \in C_k} \sum_{d=1}^{D} (X_{n_d} - \mu_{k_d})^2 \\ &= \sum_{k=1}^{K} \sum_{d=1}^{D} {X_{n_d}}^2 - \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{d=1}^{D} (SCF_{k_d})^2 \end{aligned}$$

where SCF_{k_d} is the sum of the dth feature of all voxels in C_k.

Third, in order to minimize the cross-connectivity cost, for each cortical cluster, we define the set of cortical voxels that are connected to it as its *shaded cortical cluster*.

Definition 8 (Shaded cluster). *Given a cortical partition $(C_1, \cdots, C_K)$, the shaded cluster SC_k $(k = 1, \cdots, K)$ is defined as:*

$$SC_k = \{sc | sc \in C, \forall v \in C_k, \exists v' \in C, (v', v) \in F, sc \cap C_k = \emptyset\}$$

Example 2. In Figure 2, all the cortical voxels that are connected to voxels in cortical cluster C_2 forms the shaded cluster SC_1, while all voxels that are connected to the voxles in cortical cluster C_1 forms the shaded cluster SC_2.

In an ideal coclustering, as $CoclusterB$, a shaded cluster should *coincide* with the corresponding spouse cluster. However, this is not always the case in general. The cross-connectivity cost can be characterized by the disagreement between shaded clusters and spouse clusters and quantified by the Within-Cluster Variance of shaded clusters with respect to their corresponding spouse clusters, called *Shaded Within-Cluster Variation(SWCV)*, that is defined as follows.

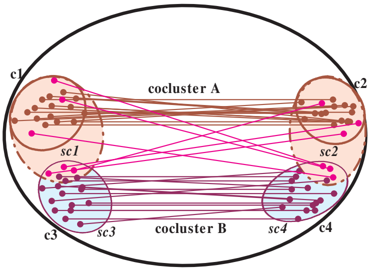


Fig. 2. Shaded clusters

Definition 9 (SWCV)**.** *The Shaded Within-Cluster Variation(*SWCV*) of cortical cluster SC_k is defined as:*

$$SWCV(SC_k) = \sum_{\overrightarrow{X'_n} \in SC_k} d(\overrightarrow{X'_n}, \overrightarrow{\mu_k})$$

Note that, instead of using the centroid of SC_k , the centroid of the C_k is used to calculate $SWCV(SC_k)$. The intuition is that, in an ideal partitioning, the shaded partition $SC_1, \cdots, SC_K$ should mostly coincide with $C_1 \cdots C_K$.

Definition 10 (STWCV)**.** *The Shaded Total Within-Cluster Variation (STWCV) of cortical partition ($SC_1, \cdots, SC_K$) is defined as:*

$$\begin{aligned} STWCV(SC_1, \cdots, SC_K) &= \sum_{k=1}^{K} SWCV(SC_k) \\ &= \sum_{k=1}^{K} \sum_{\overrightarrow{X'_n} \in SC_k} \sum_{d=1}^{D} (X'_{n_d} - \mu_{k_d})^2 \end{aligned}$$

The variance in distances between voxels is partitioned into variance attributable to differences among distance within clusters and to differences among clusters. $WCV(C_k)$ measures the variability within the cluster C_i, while we introduce $BCV(C_k)$as a measure of the variability between cortical clusters.

Statement of the problem. Finally, the coclustering problem can be formally stated as follows: given a Graph $G = (V, F)$ and a distance metric d for nodes between v_i and $v_j (i \neq j)$, coclustering is required to partition V into K clusters and cocluster sets, as well as a set of outliers, formulated as $\{< C_1, SP(C_1) >, \cdots, < C_K, SP(C_K) >, O\}$, such that the connection strength of each cluster is maximized and the following objective function $OTWCV$ is minimized:

$$OTWCV(C_1, C_2, \cdots, C_K) = \sum_{k=1}^{K} TWCV(C_k) + STWCV(SC_k)$$

4 The BCA Algorithm

In this section, we propose our coclustering algorithm, *Business model based Coclustering Algorithm (BCA)*, to solve the coclustering problem.

BCA starts with the density-based initialization, and produces a better solution from the current solution by applying the following three phases, viz. *Split*, *Transfer*, and *Merge* sequentially. Three procedures can run iteratively to produce one solution after another until a termination condition is reached. During each iteration, the current solution S_i is associated with the figure of merits that include a function of $OTWCV$ and the connection strength.

The goal of our algorithm's initialization is to not only partition cortical voxels into cocluster sets, but also to minimize the distance variance within one cluster while maximizing each cluster's connection strength.

4.1 Density-Based Initialization

The goal of our density-based initialization is to have an initial clustering of the cortical voxels based on the following working hypothesis provided by our domain experts: voxels within one functional cortical region should be close to each other and each functional cortical region should contain at least one dense subregion. The initialization procedure is described by Algorithm *Initialize* in Figure 3. The algorithm takes a cortico-cortical connectivity graph G and two parameters ε and δ as input and produces an initial coclustering as output. In addition, in the output, a set of voxels O will be identified as *outliers* that will not be classified into any functional cortical region. While ε is the maximum radius of a voxel's neighborhood, δ is the minimum number of voxels within the ε-neighborhood of a voxel for the voxel to be a core voxel. We first introduce the following notions.

Definition 11 (ε-neighborhood and core voxel)**.** *Given a cortico-cortical connectivity graph $G(V, F)$, the ε-neighborhood of a voxel $v \in V$, denoted $N_\varepsilon(v)$, is defined by $N_\varepsilon(v) = \{u \in V \mid dist(v, u) \leq \varepsilon\}$. We call v a core voxel iff $|N_\varepsilon(v)| \geq \delta$.*

Definition 12 (Distance-reachable)**.** *A voxel u is directly distance-reachable from a voxel v w.r.t. ε and δ if $u \in N_\varepsilon(v)$ and u is distance-reachable from v if there is a chain of voxels $v_1, \cdots, v_n$, such that $v_1 = v, v_n = u$, and v_k is directly distance-reachable from v_{k-1} for $k = 2, \cdots, n$.*

As shown in Figure 3, Algorithm *Initialize* firstly calculates $N * N$ *distance-connection matrix* M to store the Euclidean distances between each pair of cortical voxels (line 5). We define $dist(u, v) = +\infty$ iff u and v belong to different hemispheres of the brain to implement the constraint that each resulting cluster will not span across different hemispheres. The algorithm will iteratively consider each voxel v (lines 7 - 18). If v is an unclassified voxel, then a new cluster is formed by all the voxels that are distance-reachable from v; otherwise, either v is already classified (lines 8-9), or it is a non-core voxel (lines 14-15), the processing of v will be skipped. After the iteration completes, all the unclassified voxels will be assigned to a set O as outliers. Finally, for each identified

```
(1)  Algorithm: Initialize
(2)  Input: Cortico-cortical connectivity graph $G(V, F)$, maximum radius $\varepsilon$, and minimum number of voxels $\delta$
(3)  Output: initial coclustering $\{< C_1, SP(C_1) >, \cdots, < C_K, SP(C_K) >, O\}$
(4)  Begin
(5)      Calculate from G the distance-connection matrix $M$ to store $dist(u, v)$ for all $u, v \in V$;
(6)      $k = 0$;
(7)      For each voxel $v \in V$ do
(8)          If $v$ is classified then
(9)              Process the next voxel;
(10)         Else /* $v$ is not classified */
(11)             If $v$ is a core voxel then
(12)                 $k := k + 1$;
(13)                 Collect all voxels distance-reachable from $v$ and assign them to $C_k$
(14)             Else
(15)                 Process the next voxel;
(16)             End If
(17)         End If
(18)     End For
(19)     Collect all unclassified voxels and assign them to $O$ ;
(20)     Identify $SP(C_k)(k = 1...K)$ according to Definition 3.
(21) End Algorithm
```

Fig. 3. Algorithm Initialize

cluster C_k ($k = 1, \cdots, K$), its spouse cluster $SP(C_k)$ will be identified according to Definition 3 (line 20) to produce the initial coclustering result.

Since the analysis performed in the initialization procedure focuses on region density and distances between cortical voxels rather than their connectivity, the BCA algorithm further applies operators Split, Transfer, and Merge iteratively to improve the coclustering result by minimizing its OTWCV value.

4.2 Split

The split operator attempts to split a cluster into two clusters when such a split will improve the result of coclustering that is characterized by the following split condition.

Definition 13 (Split condition)**.** *Given a coclustering* $CO = \{< C_1, SP(C_1) >, \cdots, < C_i, SP(C_i) >, \cdots, < C_K, SP(C_K) >\}$ *and a cluster* $C_i \in CO$*, let* C_{i1} *be the set of voxels in* C_i *that are connected to* $SP(C_i)$*,* C_{i2} *be* $C_i - C_{i1}$*, and* $CO' = \{< C_1, SP(C_1) >, \cdots, < C_{i1}, SP(C_{i1}) >, < C_{i2}, SP(C_{i2}) >, \cdots, < C_K, SP(C_K) >\}$*, then we say that* C_i *satisfies the split condition iff*

1) $|C_{i1}| >= \delta$ *and* $|C_{i2}| >= \delta$*;*
2) $OTWCV(CO') \leq OTWCV(CO)$*;*
3) $\theta(C_1, SP(C_1)) \leq \theta(C_{i2}, SP(C_{i2}))$.

Intuitively, the split condition ensures that after a split, 1) the number of voxels in each new cluster is still greater than or equal to δ, 2) the OTWCV value for the new coclustering will not increase, and 3) the connection strengths of the two new clusters C_{i1} and C_{i2} will be no less than the connection strength of the original cluster C_i. This is always true for C_{i1}, and thus we only need to require $\theta(C_1, SP(C_1)) \leq \theta(C_{i2}, SP(C_{i2}))$ in the above definition of the split condition.

```
Algorithm: Split
Input: $CO = \{< C_1, SP(C_1) >, \cdots, < C_K, SP(C_K) >\}$
Output: a new version of $CO$ in which no more cluster satisfies the split condition
Begin
    While there exists a cluster $C_i \in CO$ satisfying the split condition do
        Split $C_i$ into $C_{i1}$ and $C_{i2}$;
        Recalculate the spouse cluster for each cluster in $CO$ according to Definition 3;
    End while
End Algorithm
```

Fig. 4. Algorithm Split

Algorithm *Split* is sketched in Figure 4. Basically, it iteratively splits the colustering result until no more cluster satisfies the above defined split condition.

4.3 Transfer

The transfer operator attempts to reassign each voxel to a new cluster in order to improve the result of coclustering that is characterized by the following transfer condition.

Definition 14 (Transfer Condition)**.** *Given a coclustering* $CO = \{< C_1, SP(C_1) >, \cdots, < C_i, SP(C_i) >, \cdots, < C_j, SP(C_j) >, \cdots, < C_K, SP(C_K) >\}$*, let* $v \in C_i$ *for some* C_i *in CO,* C_j *be the cluster to whose centroid* v *is the closest, after transferring* v *from* C_i *to* C_j*,* C_i *becomes* C_i'*,* C_j *becomes* C_j'*, and CO becomes* $CO' = \{< C_1, SP(C_1) >, \cdots, < C_i', SP(C_i') >, \cdots, < C_j', SP(C_j') >, \cdots, < C_K, SP(C_K) >\}$*, we say that* v *satisfies the transfer condition iff*

1) $|C_i'| >= \delta$*;*
2) $OTWCV(CO') \leq OTWCV(CO)$*;*
3) $\theta(C_i, SP(C_i)) \leq \theta(C_i', SP(C_i'))$ *and* $\theta(C_j, SP(C_j)) \leq \theta(C_j', SP(C_j'))$*.*

```
Algorithm: Transfer
Input: $CO = \{< C_1, SP(C_1) >, \cdots, < C_K, SP(C_K) >\}$
Output: new version of $CO$ in which no more voxel satisfying the transfer condition
Begin
    While there exists a voxel $v \in C_i$ satisfying the transfer condition do
        Transfer $v$ from $C_i$ to $C_j$ where $C_j$ is the cluster to whose centroid $v$ is the closest;
        Recalculate the spouse cluster for each cluster in $CO$ according to Definition 3;
        End while
End Algorithm
```

Fig. 5. Algorithm Transfer

Intuitively, the transfer condition ensures that after a transfer, 1) C_i still contains at least δ voxels, 2) the OTWCV value for the new coclustering will not increase, and 2) the connection strengths of the two affected clusters will not decrease. Algorithm *Transfer* is sketched in Figure 5. Basically, it attempts to assign each voxel to a new cluster if it satisfies the transfer condition. The procedure terminates when no more voxel satisfies the above defined transfer condition.

4.4 Merge

Finally, the merge operator attempts to merge two clusters if such a merge will improve the result of coclustering that is characterized by the following merge condition.

Definition 15 (Merge Condition)**.** *Given a coclustering* $CO = \{< C_1, SP(C_1) >, \cdots, < C_i, SP(C_i) >, \cdots, < C_j, SP(C_j) >, \cdots, < C_K, SP(C_K) >\}$*, and two clusters* $C_i, C_j \in CO$*, we merge* C_i *and* C_j *into* C_m *and derive a new coclustering* $CO' = \{< C_1, SP(C_1) >, \cdots, < C_m, SP(C_m) >, \cdots, < C_K, SP(C_K) >\}$*. We say* C_i *and* C_j *satisfy the merge condition iff*

1) $OTWCV(CO') \leq OTWCV(CO)$*;*
2) $\theta(C_i, SP(C_i)) \leq \theta(C_m, SP(C_m))$ *and* $\theta(C_j, SP(C_j)) \leq \theta(C_m, SP(C_m))$*.*

```
Algorithm: Merge
Input: $CO = \{< C_1, SP(C_1) >, \cdots, < C_K, SP(C_K) >\}$
Output: new version of $CO$ in which no more cluster satisfying the merge condition
Begin
    While there exists $C_i, C_j \in CO$ satisfying the merge condition do
        Merge $C_i$ and $C_j$ into $C_m$;
        Recalculate the spouse cluster for each cluster in $CO$ according to Definition 3;
    End while
End Algorithm
```

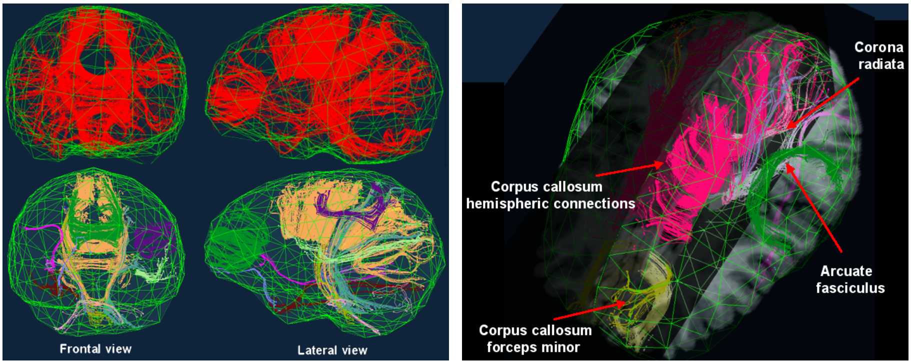
Fig. 6. Algorithm Merge

Intuitively, the merge condition ensures that after a merge, 1) the OTWCV value for the new coclustering will not increase, and 2) the connection strength of the new merged cluster is no less than the connection strengths of the two original clusters. Algorithm *Merge* is sketched in Figure 6. Basically, it merges two clusters into one if the two clusters satisfy the above defined merge condition. The algorithm terminates when no more pair of clusters satisfy the above defined merge condition.

5 3-D Visualization of the BCA Results

All fiber tracts calculated from DTI data were rendered in relation to the cortical mesh obtained from conformal brain surface mapping [18], as shown in Figure 7-(a)-Top. It can be seen that there is a large number of fiber tracts connecting cortical areas. BCA was performed based on the spatial relationship of voxels on the cortical surface and Figure 7-(a)-Bottom exhibits clustered cortical fibers in frontal and lateral view. Figure 7-(b) shows the results of our BCA in a representative subject. Well-know anatomical fiber tracts in the brain are reproduced such as the colossal fibers (pink) which connect the two hemispheres and the forceps minor of the corpus callosum (yellow) connecting the left and right side of the frontal cortex. Moreover, the intra-hemispheric connections of the arcuate fasciculus connecting Broca's and Wernicke's cortical areas can be appreciated.

These results indicate that the developed algorithm is consistent with brain anatomy and that it allows automated segmentation of the cortex based on DTI-derived cortical


Frontal view
Lateral view
Corona radiata
Corpus callosum hemispheric connections
Arcuate fasciculus
Corpus callosum forceps minor

(a) (b)

Fig. 7. (a)-Top: frontal and lateral view of cortico-cortical fibers before coclustering; (a)-Bottom: frontal and lateral view of clustered cortico-cortical connectivity; (b) Zoom in view of some specific coritco-cortical clusters

connections within the brain. We therefore believe that our algorithm is well suited to provide an efficient framework for further analysis including the quantitative assessment of cortico-cortical connectivity.

6 Conclusions and Future Work

In this paper, we defined the coclustering problem and we applied this approach to the analysis of cortico-cortical connections in the brain. Our approach represents an efficient mathematical framework that is computationally robust and is able to be used for quantitative analysis of cortico-cortical fiber tracts. This in turn might be relevant for the identification of secondary epileptic foci in patients with intractable epilepsy and might impact their clinical management.

Although the coclustering problem was initially motivated by the need of cortico-cortical connectivity analysis, we expect that it will have a wide range of applications. In the future, we plan to apply our BCA also to the analysis of thalamo-cortical connectivity and the segmentation of thalamic nuclei.

Acknowledgment

This research was partially supported by the Michigan Technology Tri-Corridor basic research grant MTTC05-135/GR686.

References

1. D. Le Bihan, E. Breton, D. Lallemand, P. Grenier, E. Cabanis and M. Laval-Jeantet. MR imaging of intravoxel incoherent motions: application to diffusion and perfusion in neurologic disorders. *Radiology*, 161:401–407, 1986.

2. D. Le Bihan. Looking into the functional architecture of the brain with diffusion MRI. *Nature Rev Neurosci*, 4:469–480, 2003.
3. P.J. Basser and C. Pierpaoli. Microstructural and physiological features of tissues elucidated by quantitative diffusion-tensor MRI. *J Magn Reson*, Series B:209 – 219, 1996.
4. A.W. Toga P.M. Thompson. A framework for computational anatomy. *Computing and Visualization in Science*, 5:13–34, 2002.
5. R.E. Passingham , K.E. Stephan and R. Kotter. The anatomical basis of functional localization in the cortex. *Nat Rev Neurosci*, 3:606–616, 2002.
6. J. Han and M. Kamber. *Data Mining*, pages 349–353. I-55860-489-8. Morgan Kaufmann Publishers, 340 Pine Street, Sixth Floor, San Francisco, CA 94104-3205, USA, 2001.
7. R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *20th Int. Conf. on Very Large DataBases*, pages 144–155, 1994.
8. S. Guha, R. Rastogi and K. Shim. Cure: An efficient clustering algorithm for large databases. *ACM SIGMOD international conference on management of data*, pages 73–84, 1998.
9. M. Ester, H.P. Kriegel, J. Sander and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. *2nd Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.
10. T.E. Conturo , N.F. Lori, T.S. Cull, E. Akbudak, A.Z. Snyder, J.S. Shimony, R.C. Mckinstry, H. Burton and M.E. Raichle. Tracking neuronal fiber pathways in the living human brain. *Natl. Acad. Sci. USA*, 96:10422–10427, 1999.
11. R.F. Dougherty, M. Ben-Shachar, R. Bammer, A.A. Brewer and B.A. Wandell. Functional organization of human occipital-callosal fiber tracts. *Natl. Acad. Sci. USA*, 102:7350–7355, 2005.
12. P.J. Basser , S. Pajevic, C. Pierpaoli, J. Duda and A. Aldroubi. In vivo fiber tractography using DT-MRI data. *Magn Reson Med*, 4:625–32, 2000.
13. S. Mori, B.J. Crain, V.P. Chacko and P.C.M. van Zijl. Three dimensional tracking of axonal projection in the brain by magnetic resonance imaging. *Ann. Neurol.*, 45:265–269, 1999.
14. G. Karypis, E. Han and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. *Computer*, 32(8):68–75, 1999.
15. M. Ankerst, M.M. Breunig, H. Kriegel and J. Sander. Optics: ordering points to identify the clustering structure. *ACM SIGMOD international conference on management of data*, pages 49 – 60, 1999.
16. A. Hinneburg and D.A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Knowledge Discovery and Data Mining*, pages 58–65, 1998.
17. C. Lin, S. Lu, X. Liang, and J. Hua. GCA: a Coclustering Algorithm for Thalamo-Cortico-Thalamic Connectivity Analysis. *ICDM Workshop on Data Mining in Bioinformatics*, pages 163–168, December 2006.
18. G. Zou, J. Hua, X. Gu, and O. Muzik. An Approach for Intersubject Analysis of 3D Brain Images based on Conformal Geometry. *IEEE Int. Conf. on Image Processing*, pages 1193 – 1196, 2006.

An Algorithm for Hierarchical Classification of Genes of Prokaryotic Genomes

Hongwei Wu*, Fenglou Mao*, Victor Olman, and Ying Xu**

Computational Systems Biology Laboratory
Department of Biochemistry and Molecular Biology and Institute of Bioinformatics
University of Georgia, Athens, GA30602
{hongweiw@csbl,fenglou@csbl.,olman@csbl.,xyn@}bmb.uga.edu

Abstract. We present in this paper our hierarchical classification of genes for prokaryotic genomes from a methodological point of view. Our classification scheme is unique in that (1) the functional equivalence relationships among genes are assessed by using both sequence similarity and genomic context information, (2) genes are grouped into clusters of multiple resolution levels based on their equivalence relationships among each other, and (3) gene clusters, which are either *parallel-to* or *inside-of* one another, naturally form a hierarchical structure. This classification scheme has been applied for the genes of 224 complete prokaryotic genomes (release as of March, 2005). The classification results are available at http://csbl.bmb.uga.edu/HCG, and are validated through comparisons with the taxonomy of these 224 genomes, and with two existing gene classification schemes, Clusters of Orthologous Groups of proteins (COG) and Pfam, respectively.

1 Introduction

Functional classification of genes (or gene products) has been generally done under the framework of homology and orthology, where homologous genes are considered to be functionally equivalent at a coarse level and orthologous genes are considered to be equivalent at a fine level. As our knowledge about gene functions accumulates, it has become clear that the concepts of homology/orthology, which comes from studies of gene evolution, may not necessarily be the most appropriate concepts for functional classification of genes, particularly at high resolution levels. For example, the L-serine dehydratase, which converts serine into pyruvate in the gluconeogenesis pathway, is encoded by a single gene, *sda*, in *Escherichia coli*, but is encoded by two separate genes, *sdhA* and *sdhB*, in *Bacillus subtillis*. Also, because in some species (e.g., *E. coli* and *Bacillus anthracis*) there exist multiple mechanisms to synthesize L-serine dehydratese, the *sda*, *sdhA* and *sdhB* genes each may correspond to multiple versions (e.g., *sdaA*/*sdaB* for *sda*, *sdhA-1*/*sdhA-2* for *sdhA*, and *sdhB-1*/*sdhB-2* for *sdhB*, respectively) [1]. At a coarse level, all these L-serine dehydratase genes, *sda*/*sdaA*/*sdaB*, *sdhA*/*sdhA-1*/*sdhA-2*, *sdhB*/*sdhB-1*/*sdhB-2*, can be viewed as functionally equivalent; at

[*] These two authors have contributed equally to this paper.
[**] Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 551–563, 2007.

a finer level, however, the *sda*/*sdaA*/*sdaB*, *sdhA*/*sdhA-1*/*sdhA-2* and *sdhB*/*sdhB-1*/*sdhB-2* genes can be grouped into three different subgroups; and at a even finer level, the different versions of the same gene (e.g., *sdaA* and *sdaB* for *sda*, *sdhA-1* and *sdhA-2* for *sdhA*, and *sdhB-1* and *sdhB-2* for *sdhB*) can be grouped into different sub-sub-groups. This example indicates that the functional equivalence relationships among genes could be multi-layered, and a new classification framework that is more general than homology/orthology may be needed to better capture these relationships at finer resolution levels.

We have developed a computational framework for *h*ierarchical *c*lassification of *g*enes (HCG) for prokaryotic genomes to capture the functional equivalence relationships among genes at multiple resolution levels, and have applied this scheme to 224 complete prokaryotic genomes (ftp://ftp.ncbi.nih.gov/genomes/Bacteria/, March 2005)[1]. We have validated the HCG in a systematic manner by comparing the classification results for these 224 genomes with the taxonomy of the involved genomes, and with two of the existing classification systems of genes, Cluster of Orthologous Groups of proteins (COG) [4, 5] and Pfam [6]. Due to the limit of space, we only focus on the methodological issues of the HCG in the paper. More details about the results and validations are provided in [7].

2 Method

Our HCG scheme consists of three steps. First, we define a measure to capture the functional equivalence relationship between a pair of genes based on both their sequence similarity and conserved genomic neighborhood information. Secondly, we apply this scoring function to all gene pairs under study, and apply a minimum spanning tree (MST)-based clustering algorithm on the graph representation of genes and their equivalence relationships to identify clusters. In the final step, we apply an annotation algorithm to summarize the functional properties of each cluster. However, due to the limit of space, we can only focus on the first two steps but provide a brief description of the final step.

In the rest of this paper, we only consider those gene pairs whose reciprocal BLASTP [8] e-values ≤ 1.0.

2.1 Functional Equivalence Relationship Between Genes

2.1.1 Training Sets

Before explaining how to measure the functional equivalence relationships among genes, we first describe the positive and negative training sets based on which those parameters used for the assessments are optimized. A pair of genes from two different genomes is considered to be *orthologous* and is therefore included in the positive training set if and only if the two genes are both enzymes with identical EC numbers and exactly the same enzymatic property descriptions (http://ca.expasy.org/enzyme/) [9]. Whereas, a pair of genes from two different genomes, ($\hat{g}_1$, $\hat{g}_2$), is considered to be *non-orthologous* and is therefore included in the negative training set if and only if

[1] The classification results are stored in a database at http://csbl.bmb.uga.edu/HCG

there is an orthologous pair ($\hat{g}_1$, g_2) in the positive set such that (i) g_2 and $\hat{g}_2$ correspond to different genes of the same genome, and (ii) the probability for g_2 and $\hat{g}_2$ to be in the same operon is negligible (See section 2.1.4 for more discussion on operons), suggesting little chance for them to be duplicated genes.

2.1.2 Measure of the Functional Equivalence

In prokaryotic genomes, genes belonging to the same genomic neighborhood (e.g., operons) are usually functionally related; hence, it is possible to incorporate so-encoded functional information of genes to better quantify the functional equivalence relationships among genes. We have recently observed that about 20% of orthologous gene pairs but only about 3% of non-orthologous gene pairs in the training sets are accompanied by gene pairs in their genomic neighborhoods[2], indicating that gene pairs with companions are more likely to be orthologous than non-orthologous.

Inspired by this observation, we have integrated the sequence similarity information and the genomic neighborhood information, as in (1), to quantify the functional equivalence relationship between genes.

$$f(g_1,g_2)=h(g_1,g_2)\left[1+\lambda\sum_{i,j}P((g_1,g_i)\in \text{Operon})P((g_2,g_j)\in \text{Operon})I(h(g_i,g_j)\geq t_h)\right] \quad (1)$$

In this definition, $h(\cdot,\cdot)$ represents the gene pair's sequence similarity measure – the larger $h(\cdot,\cdot)$ is, the more similar two genes are from the sequence point of view (See Section 2.1.3); the genomic neighborhood of a gene g consists of those genes g_i whose probabilities to be in the same operon as g are non-negligible (See Section 2.1.4); the summation $\Sigma_{i,j}$ is over all the gene pairs (g_i, g_j) with g_i and g_j belonging to the genomic neighborhoods of g_1 and g_2, respectively; $P((\cdot,\cdot)\in \text{Operon})$ is the probability for two genes to belong to the same operon; $I(\cdot)$ is an indicator function; t_h is the threshold so that a pair of genes with their $h(\cdot,\cdot)$ value above t_h is more likely to be orthologous than non-orthologous from the sequence similarity point of view; and λ determines the level of contribution from the gene pair's neighborhood information to their final equivalence measure $f(\cdot,\cdot)$.

Before we describe how $h(\cdot,\cdot)$, t_h, $P((\cdot,\cdot)\in \text{Operon})$ and λ are estimated in the following sub-sections, we first explain why the particular form of (1) is used. First, we believe that a gene pair's sequence similarity information should be a dominating factor in determining the level of functional equivalence between the two genes, while their genomic neighborhood information, which only suggests the possibility of the gene pair's functional equivalence, should play only a secondary role. Secondly, the more likely two genes belong to the same operon, the more functionally related the two genes are. Therefore, we have set the impact of (g_i, g_j) on the functional equivalence relationship between g_1 and g_2 proportional to the probability that g_1 and g_i (as well as g_2 and g_j) belong to the same operon. Finally, for a gene pair (g_1, g_2), the gene pairs in their genomic neighborhoods should be reliable enough to be considered as supporting evidence for (g_1, g_2)'s functional equivalence relationship. Therefore,

[2] A gene g_m is considered to be in the genomic neighborhood of another gene g_n if and only if the probability for g_m and g_n to belong to the same operon is non-negligible. See section 2.1.4 for more discussions.

by using the indicator function and t_h, only those pairs that are more likely to be orthologous than non-orthologous are incorporated.

2.1.3 Sequence Similarity Measure $h(\cdot,\cdot)$ and Threshold t_h

It is known that the BLAST e-value depends on the search space, and it can be different for the two directions of the alignment for the same pair of sequences. This suggests that the BLAST e-values alone may not represent the most reliable measure for sequence similarities. We have therefore turned to the slower but more reliable Smith-Waterman (SW) algorithm [10, 11]. Given a pair of genes of two different genomes, (g_1, g_2), we have collected the five SW measurements, including the SW score x_1, the length of the aligned (sub-)sequence x_2, the percentages of sequence identity x_3, the percentage of sequence similarity x_4, and the percentage of sequence gap x_5. By combining these five SW measurements plus the sequence lengths of g_1 and g_2 (l_1 and l_2, respectively) as in (2), we have obtained $h(\cdot,\cdot)$ as a measure for the sequence similarity between genes .

$$h(g_1,g_2) \equiv h(x_1,x_2,x_3,x_4,x_5,l_1,l_2|(g_1,g_2)) = \frac{a_1x_1}{x_2^{\gamma}} + \frac{a_2x_2}{l_1+l_2} + a_3x_3 + a_4x_4 + a_5x_5 \quad (2)$$

Note that $h(\cdot,\cdot)$ can actually be interpreted as a linear combination of the five SW measurements, except for that a_1x_1/x_2^{γ} normalizes the SW score x_1 with the length of the aligned subsequence x_2. Introducing the parameter γ in a_1x_1/x_2^{γ} allows flexibility of normalization so that a_1x_1/x_2^{γ} can faithfully reflect the quality of sequence alignment regardless of the lengths of the genes. There are certainly many other ways to combine these different SW measures. The main reason that drives us to use (2) is its simplicity and efficiency. After the parameters a_i ($i=1, \ldots, 5$) and γ are optimized to best discriminate between the positive and negative training sets with a Bayesian classifier, we have obtained the classification error rate as 13.12%. As a comparison, when a Bayesian classification is performed directly on BLASTP e-values, we have obtained the classification error rate as 18.47%. This indicates that $h(\cdot,\cdot)$, into which the multi-dimensional SW measures is fused in even just a simple fashion, is better than BLASTP e-values in characterizing the sequence similarity between genes.

With the parameters of (2) being optimized, a gene pair with $h(\, , \,) \geq t_h$=2.984 is more likely to be othorlogous than non-orthologous just from the sequence similarity point of view.

2.1.4 Probability of Two Genes Belonging to Same Operon, $P((\cdot,\cdot) \in \text{Operon})$

We assume that whether two adjacent genes on the same strand, g_i and g_{i+1}, belong to the same operon only depends on g_i and g_{i+1} themselves but not on any other genes. Then, given two genes g_m and g_n on the same strand without intervening genes on the opposite strand, $P((g_m, g_n) \in \text{Operon})$ can be computed as:

$$P((g_m,g_n) \in \mathit{Operon}|\text{features}) = \prod_{i=m}^{n-1} P((g_i,g_{i+1}) \in \mathit{Operon}|\text{features}) \quad (3)$$

Note that *features* in (3) refer to the features used for operon prediction, and usually include one or more of the following characteristics: (i) the inter-genic distance between the two genes, (ii) whether the two genes have co-evolved, (iii) whether the two genes belong to the same conserved genomic neighborhood, and (iv) whether the two genes are functionally related.

Note that some of the above features, especially the second and third ones, are usually obtained through the identification of orthologous genes. Since we cannot assume the availability of orthology mapping or functional annotations of genes, we have only used the inter-genic distance as the feature. By using the Bayesian rule, the probability of g_i and g_{i+1} belonging to the same operon provided their inter-genic distance $d_{i,i+1}$ can be computed as:

$$P\left((g_i, g_{i+1}) \in O \middle| d_{i,i+1}\right) = \frac{P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \in O\right) P\left((g_i, g_{i+1}) \in O\right)}{P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \in O\right) P\left((g_i, g_{i+1}) \in O\right) + P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \notin O\right) P\left((g_i, g_{i+1}) \notin O\right)} \tag{4}$$

where O stands for operon, $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \in O\right)$ [or $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \notin O\right)$] describes the conditional distribution of $d_{i,i+1}$ when g_i and g_{i+1} do (or do not) belong to the same operon, and $P\left((g_i, g_{i+1}) \in O\right)$ [or $P\left((g_i, g_{i+1}) \notin O\right)$] is the a-priori probability that any two adjacent genes on the same strand do (or do not) belong to the same operon.

To estimate the two *a-priori* probabilities, $P\left((g_i, g_{i+1}) \notin O\right)$ and $P\left((g_i, g_{i+1}) \in O\right)$, we have applied the assumption used in [12] that the adjacent two genes not belonging to the same operon are equally likely to be on the same strand or on opposite strands. Under this assumption, Ermolaeva et al. have suggested that $P\left((g_i, g_{i+1}) \notin O\right)$ can be estimated as the ratio of the number of directons[3] to the number of gene pairs whose two component genes are adjacent on the same strand [12]. For the genomes covered by the four different operon prediction methods [12-15], two adjacent genes on the same strand are more likely to belong to the same operon than not, 58% vs. 42%, before we start to consider any additional information.

To estimate the two conditional probabilities $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \in O\right)$ and $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \notin O\right)$, we have utilized the operon prediction results provided by [12-15] to construct two data sets, $\left\{(g_i, g_{i+1}) \in O\right\}$ and $\left\{(g_i, g_{i+1}) \notin O\right\}$. The set $\left\{(g_i, g_{i+1}) \in O\right\}$ consists of gene pairs whose two component genes are adjacent on the same strand and are predicted to belong to the same operon by at least one of the four prediction methods [12-15]. For the $\left\{(g_i, g_{i+1}) \notin O\right\}$ set, by applying the assumption used in [12] that the distribution of $d_{i,i+1}$ is the same for all non-operonic pairs, regardless whether the adjacent two genes are on the same or the opposite strands, we have included those gene pairs whose two component genes are adjacent on the opposite strands. Since $d_{i,i+1}$ ranges from -5,951 to 12,991 bps for the $\left\{(g_i, g_{i+1}) \in O\right\}$ set and from -12,853 to 50,476 bps for the $\left\{(g_i, g_{i+1}) \notin O\right\}$ set, it would be impractical to estimate $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \in O\right)$ and $P\left(d_{i,i+1} \middle| (g_i, g_{i+1}) \notin O\right)$ for every possible value of $d_{i,i+1}$. Instead, we have applied a discretization method adapted from [16] to segment the entire range of $d_{i,i+1}$ into small intervals $\{I_1, I_2, \ldots\}$, and have then estimated

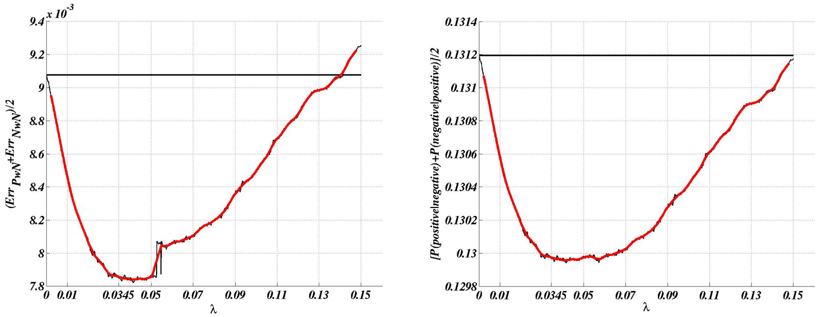
[3] A *directon* is a list of genes arranged in tandem in the same strand of the genome without being interrupted by genes on the other strand.

$P\left(d_{i,i+1} \in I_k \middle| (g_i, g_{i+1}) \in O\right)$ and $P\left(d_{i,i+1} \in I_k \middle| (g_i, g_{i+1}) \notin O\right)$ for each such small interval. After excluding some unreliable estimations, we have obtained $P\left((g_i, g_{i+1}) \in O \middle| d_{i,i+1} \in I_k\right)$ as *non-negligible* when $d_{i,i+1}$ belongs to [-13, 151] bps.

A gene g_m is considered to be in the genomic neighborhood of another gene g_n if and only if the probability for g_m and g_n to belong to the same operon, as computed in (3), is non-negligible.

2.1.5. Estimation of λ

Given a gene pair (g_1, g_2), if there exist gene pairs, (g_i, g_j), with g_i and g_j belonging to g_1 and g_2's genomic neighborhoods, respectively, and $h(g_i, g_j) \geq t_h$, then (g_1, g_2) is viewed to be *with neighborhood confirmations*; otherwise (g_1, g_2) is viewed to be *without neighborhood confirmations*. When both the *positive/negative* and *with/without neighborhood confirmations* are considered, the training data can be divided into four different subsets, namely, *positive with neighborhood confirmations* (PwN), *positive without neighborhood confirmations* (PwoN), *negative with neighborhood confirmations* (NwN), and *negative without neighborhood confirmations* (NwoN), respectively. Because it is only for the PwN and NwN data that the *f*(,) value is affected by different values of λ, we have chosen λ so as to optimally discriminate between these two subsets.



(a) (b)

Fig. 1. (a) (Err_{PwN}+Err_{NwN})/2 and (b) [*P*(*positive*| *negative*)+*P*(*negative*| *positive*)]/2, as functions of λ. In both figures, the black straight lines correspond to what are obtained when $\lambda=0$, and the black fluctuated curves correspond to what are obtained by varying λ from 0 to 0.15 with step-size of 0.0001, and the red curves are obtained by smoothing the black fluctuated curves with a 45-point rectangular window.

Consider that a Bayesian classifier is applied to *f*(,) for distinguishing between the positive and negative training data. Let *P*(*negative*|*positive*) denote the probability of incorrectly classifying a positive data as negative, *P*(*positive*|*negative*) denote the probability of incorrectly classifying a negative data as positive, Err_{PwN} denote PwN's contribution to *P*(*negative*|*positive*), and Err_{NwN} denote NwN's contribution to *P*(*positive*|*negative*). As the value of λ varies, the *f*(,) value for both PwN and NwN also varies; hence, the distributions of *f*(,) for both the positive and negative sets, the classification errors *P*(*negative*|*positive*) and *P*(*positive*|*negative*), and the error contributions from PwN and NwN sets, Err_{PwN} and Err_{NwN}, all vary as well. Fig. 1

shows $[P(negative|positive)+P(positive|negative)]/2$ and $(Err_{PwN}+Err_{NwN})/2$ as functions of λ. Observe from the figure that both $(Err_{PwN}+Err_{NwN})/2$ and $[P(negative|\ positive)+ P(positive|\ negative)]/2$ remain in relatively flat valleys when λ is in the interval of [0.03, 0.05]. Therefore, without loss of generality, we have chosen λ to be 0.0345.

2.2 Hierarchical Clustering of Genes

Genes and their functional equivalence relationships among each other can be represented by a weighted graph $G(V^{all}, E^{all})$, where V^{all} and E^{all} denote the node set and edge set, respectively, As exemplified by Fig. 2 (a), in this graph representation, each node $\in V^{all}$ represents a gene, and the weight on each edge equals to the $f(\ ,\ )$ value between the two genes being connected. In the rest of this paper, we use the two terms, (sub-)*graph* and *cluster*, inter-changeably.

Note that $G(V^{all}, E^{all})$ should ideally consist of a number of unconnected sub-graphs with each sub-graph representing a group of genes that are functionally equivalent at a coarse level. However, when $G(V^{all}, E^{all})$ is to represent the functional equivalence relationships among a huge number of genes (e.g., 658,174 genes from 224 genomes in our study), there inevitably exist some coincidental edges whose $f(\ ,\ )$ values are insignificant and whose existences cannot be confirmed by other edges. Therefore, we have applied a graph partitioning algorithm, the Markov Cluster Algorithm (MCL) [17], to remove these coincidental edges. The resulting sub-graphs, $G(V^i, E^i)$ (i=1, 2, …), each can be viewed to represent a group of genes that are functionally equivalent in a loose sense. Fig. 2 (a) shows one such sub-graph. Observe from the figure that even within the same sub-graph some genes are more equivalent with each other than they are with the other genes and therefore form a densely intra-connected sub-sub-graph. Note that these densely intra-connected sub-sub-graphs naturally form a hierarchical structure, and this hierarchy of clusters strongly suggests the need for a multi-level clustering scheme. We present here one such clustering scheme.

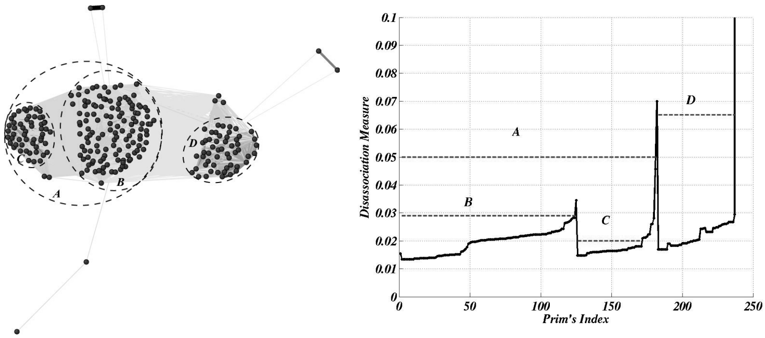


Fig. 2. (a) The graph representation for a group of genes that are functionally equivalent at a coarse level, where genes inside each ellipse are more equivalent to each other than they are to the genes outside. The layout of the nodes and edges is generated by using the Pajek Software (http://vlado.fmf.uni-lj.si/pub/networks/pajek/), where the Euclidean distance between two genes and the darkness of their connecting edge are both roughly proportional to their $f(\ ,\ )$ value. (b) The sequential representation of the graph in (a) that is obtained during our MST-based clustering. Ellipses in (a) have one-to-one correspondences with the valleys with identical labels in (b).

2.2.1 Minimum Spanning Tree (MST) Based Clustering Algorithm

Our clustering algorithm is based on a minimum spanning tree (MST) representation of a graph. The same idea has been successfully used in the identification of regulatory binding motifs [3], clustering gene expression data [2], etc. The characteristics of this clustering algorithm are summarized as follows:

A. Disassociation Measure: Note that the level of functional equivalence between a pair of genes, (g_1, g_2), is reflected not only by their own $f(g_1, g_2)$ value but also by the equivalence measures between g_1 (or g_2) with the other genes g_k ($k\neq 1$ or 2), $f(g_1, g_k)$ [or $f(g_2, g_k)$]. Therefore, by combining their own equivalence measure with the confirmative evidence from the additional genes g_k ($k\neq 1$ or 2), we have defined the *disassociation* measure for (g_1, g_2), $d(g_1, g_2)$, as:

$$d(g_1,g_2)=\left(f^2(g_1,g_2)+\frac{\rho}{r}\sum_{k=1}^{r} f\left(g_1,g_{(k)}\right)f\left(g_{(k)},g_2\right)\right)^{-1} \qquad (5)$$

where $g_{(k)}$ is the k-th ranked additional gene in terms of the value of $f(g_1, g_k)f(g_2, g_k)$, r is the maximum number of the additional genes allowed to be counted, and ρ determines the level of confirmative evidence that the additional genes adds. The two parameters r and ρ provide flexibilities for a user to tailor the clustering method to their specific problems. In our study, ρ is set to be 0.6, and r is set adaptive to the number of genes included in $G(V^i, E^i)$ with some saturation effect, as $r=\left\lfloor \mu_0\left|V^i\right| / \left(\left|V^i\right|+\mu_1\right)\right\rfloor$ with $|\cdot|$ representing the cardinality and $\lfloor\ \rfloor$ representing the *floor* function, and μ_0 and μ_1 being set as 40 and 100, respectively.

B. Cluster Identification: A *spanning tree* [18] of a graph $G(V, E)$ is a connected sub-graph that includes all nodes of V but does not contain any cycle, while a *minimum spanning tree* (MST) is a spanning tree with the minimum total edge-distance (disassociation) measure. Any connected component of a MST is called a *sub-tree* of the MST. Based on the definition of a cluster in [2, 3], for which, the disassociation measures within a cluster should be smaller than inter-cluster disassociation measures, the clustering problem can be reduced to a tree partitioning problem and can then be solved in the following two steps:

- By applying Prim's algorithm [19] to construct an MST of $G(V, E)$, we can map $G(V, E)$ to a list of nodes, $\{g_{(1)} \ldots g_{(|V|)}\}$, with $g_{(j)} \in V$ being the node selected at the j-th step. When $g_{(j)}$ gets selected, the disassociation measure between $g_{(j)}$ and the gene that recruits $g_{(j)}$ into the current MST, $\min_{1\leq i\leq j-1}\{d(g_{(i)}, g_{(j)})\}$, is the smallest among all possible candidates to be selected. We call $\min_{1\leq i\leq j-1}\{d(g_{(i)}, g_{(j)})\}$ as the dissociation measure on $g_{(j)}$'s *selection edge*, and the list $\{g_{(1)} \ldots g_{(|V|)}\}$ along with the disassociation measures on the corresponding selection edges as the *sequential representation* of $G(V, E)$.
- We represent the information of a sequential representation of $G(V, E)$ by using a two-dimensional plot, as in Fig. 2(b), where along the horizontal axis are $g_{(1)} \ldots g_{(|V|)}$ and along the vertical axis are the disassociation measures on the corresponding selection edges. Based on our previous work [2, 3], each *valley* in this plot, which is a sub-list of nodes whose internal disassociation measures are smaller than their disassociation measures with the outside nodes, renders a cluster. Additionally, because of the hierarchical nature of valleys, it is

straightforward to identify hierarchical clusters by searching for (sub-)valleys within (super-)valleys.

C. Statistical Significance Assessment of Clusters: Given a sequential representation of a graph $G(V, E)$, if there is no cluster structure, then the disassociation measures on the nodes' selection edges comply to the Dirichlet distribution [20]. So, the statistical significance for a subset of nodes to form a cluster is reflected by the p-value computed for the hypothesis that the disassociation measures on these nodes' selection edges comply to the Dirichlet distribution. The smaller the p-value is, the less likely it is that these disassociation values comply to the Dirichlet distribution, and therefore the more statistical significant the cluster is. Given a sub-set of nodes $\{g_{(m)}, \ldots, g_{(n)}\}$ that are likely to form a cluster, we have used the following ratio as the statistic to test the hypothesis that the disassociation measures on the selection edges of $\{g_{(m)}, \ldots, g_{(n)}\}$ comply to the Dirichlet distribution.

$$T = \frac{\max_{m+1 \le j \le n} \min_{1 \le i \le j-1} d\left(g_{(i)}, g_{(j)}\right)}{\min\left\{\min_{1 \le i \le m-1} d\left(g_{(i)}, g_{(m)}\right), \min_{1 \le i \le n} d\left(g_{(i)}, g_{(n+1)}\right)\right\}} \tag{6}$$

The details for cluster identification and statistical significance assessment of clusters can be found in [3].

2.2.2 Cluster Pruning Based on the Taxonomy of the Prokaryotic Genomes

It is non-trivial to choose a threshold for T in (6) (or equivalently a threshold for the p-value) that is appropriate for all clusters. If the threshold for T is set too small, then only those mathematically very dominant clusters can be detected. In order not to miss too many of those clusters that are mathematically less dominant but may be of great biological meanings, we have first used a fairly high threshold, 0.95, for T (which is that a group of nodes are considered to form a cluster mathematically if their inside disassociation measures are no greater than 0.95 of their disassociation measures with the outside); and, have then utilized the taxonomy of the prokaryotic genomes to perform some pruning.

Prokaryotic genomes are classified in a hierarchical way, based on their ribosomal RNA genes and their morphological and physiological characteristics [21, 22] (http://www.ncbi.nlm.nih.gov/Taxonomy/taxonomyhome.html/). The resulting taxonomic lineages are organized in a tree, which, from the root (the most general) level down to the leaf (the most specific) level, consists of *superkingdom* (SK), *phylum*, *class*, *order*, *family*, *genus*, and *species*, and can roughly reflect the relative evolutionary distances among genomes.

We have used two measures to quantify the taxonomic commonality of a group of genomes. One, denoted as MSCTL, is the level of the most specific taxonomic lineage that is common to all the genomes in the group; and the other, denoted as MCTL and defined in (7), is the level of the taxonomic lineage that best describes the genomes in the group.

$$MCTL(G_1, \cdots, G_m) = \arg\max_{\forall t} \left(\frac{\left|Genomes(t) \cap \{G_1, \cdots, G_m\}\right|}{\left|Genomes(t) \cup \{G_1, \cdots, G_m\}\right|} \right) \tag{7}$$

In (7), $\{G_1, \ldots, G_m\}$ represents a group of genomes, t denotes a taxonomic lineage, the maximization is over all possible lineages in the taxonomic hierarchy, *Genomes*(t)

refers to the genomes whose taxonomic lineages include t, $\cap$ and $\cup$ denote the intersection and union operations, respectively, and $|\cdot|$ denotes the cardinality of a set. Both the MSCTL and MCTL measures can be used to reveal the taxonomic diversity of the genomes covered by a cluster. The more specific the MSCTL (or MCTL) measure is, the less diversified the genomes are covered by a cluster.

Cluster pruning is performed based on the MSCTL and MCTL values of the clusters. The basic idea is that a cluster is kept if and only if its relationship with its parent cluster reflects a child-parent relationship from the taxonomic point of view. More specifically, a cluster $G(V, E)$ is kept if and only if it satisfies one of the following three conditions: (1) $G(V, E)$ corresponds to a root level cluster, (2) $G(V, E)$ does not contain multiple genes of the same genome, while $G(V, E)$'s parent cluster does, and (3) either the MSCTL or the MCTL measure of $G(V, E)$ is strictly more specific than the corresponding measure of $G(V, E)$'s parent cluster.

2.3 Functional Annotations of the HCG Clusters

For each cluster in the HCG, we have selected a non-redundant set of NCBI annotations as well as a non-redundant set of Gene Ontology (GO) annotations [23] (http://www.ebi.ac.uk/GOA/) that are frequently used for the genes belonging to this cluster to describe the cluster's functional properties.

3 Results and Discussion

By applying our HCG scheme to 224 complete prokaryotic genomes, we have obtained 51,205 clusters covering 609,887 (~92.7%) of the total 658,174 genes, and have deposited these results into a database (http://csbl.bmb.uga.edu/HCG/). These clusters are organized into 5,339 multi-leveled and 15,770 single-leveled non-overlapping trees, where the multi-leveled trees totally contain 35,435 clusters covering 534,818 (~87.7%) genes, and the single-leveled trees totally contain 15,770 clusters covering 75,067 (~12.3%) genes. In a multi-leveled cluster tree, each child cluster only contains a subset of genes of its parent cluster, and sibling clusters do not have overlap. The number of genes included in each multi-leveled tree ranges from five to 3,938 with the average being 100.2, the number of genomes covered by each multi-leveled tree ranges from three to 224 with the average being 52.3, the number of clusters in each multi-leveled tree ranges from two to 203 with the average being 6.64, and the tree depth of a multi-leveled tree ranges from two to 10 with the average being 3.41.

In [7], we have shown through examples that the HCG can capture the functional equivalence relationships among genes at different levels, which are more consistent with the biochemical descriptions of genes that are already documented in literatures; and that the HCG can also be used to reveal the evolutionary traces of genes/genomes. However, due to the limit of space, we cannot present similar examples here, but can only briefly describe the computational validations of our clustering results, which are conducted through comparisons of the HCG results with the taxonomic hierarchy of prokaryotic genomes, and with two of the existing systems for the classification of genes, COG [5, 24] and Pfam [6].

3.1 Comparison with Taxonomic Hierarchy of Genomes:

The taxonomy of prokaryotic genomes roughly reflects whether two genomes are evolutionarily close or distant. For a pair of genes, we have used the taxonomic lineages of their pertinent genomes to measure their taxonomic distance. As shown in Fig. 3, compared to a pair of genes that are randomly picked out of the pool consisting of all the genes of the 224 genomes, a pair of genes in the same cluster at the leaf level of the HCG are much more likely to belong to the same genome, species, genus or family, but are much less likely to belong to different phyla or different superkingdoms. In contrast, a pair of genes in the same cluster at the root level of the HCG does not have any preference to any taxonomic level; and a pair of genes in the same cluster at the middle level of the HCG represents a transition from the distribution of the leaf-levels to the distribution of the root-levels. The trend that the taxonomic diversity of each cluster varies along different levels of the cluster hierarchy is consistent with the taxonomic hierarchy of the involved genomes, which means that our clustering results can be used to reveal evolutionary trace of genes/genomes.

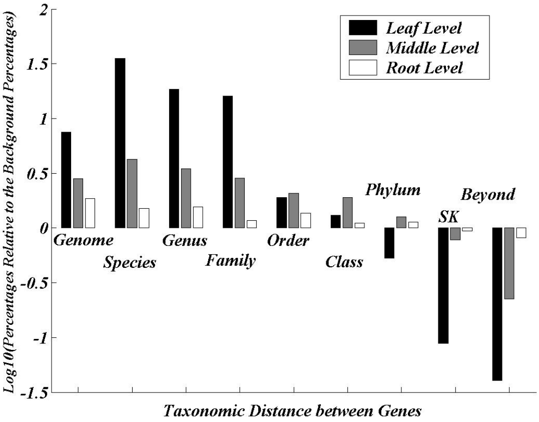


Fig. 3. Distribution of the taxonomic distance of a pair of genes belonging to the same cluster at the root, middle and leaf levels of the HCG, relative to the distribution of the taxonomic distance of a pair of genes that are randomly picked out of the whole gene pool. SK stands for super-kingdom, and *beyond* means that two genes do not even belong to the same super-kingdom. Each bin represents the ratio of the percentage of the gene pairs at a particular level of the HCG to the percentage of the gene pairs that are randomly picked out of the whole gene pool, where these gene pairs of interest all have the same level of taxonomic distance.

3.2 Comparisons with COG and Pfam Classifications

We have used the Jaccard's coefficient to measure the consistency between two clusters, and have considered two clusters *matched* if their consistency measure is above the threshold MD_0=2/3. About 85.3% of the COG clusters each can be matched by one of the HCG clusters, and about 72.5% of the Pfam clusters each can be matched by one of the HCG clusters. In contrast, only about 55.90% of the COG clusters each can be matched by one of the Pfam clusters, and about 44.74% Pfam clusters each can be matched by one of the COG clusters. The consistency measures between the HCG and the COG and Pfam, in the context of the consistency measures between the COG and Pfam, demonstrate that the information conveyable through these two existing systems, including the clusterabilities of genes and difference among different gene clusters, can essentially be conveyed by the HCG.

We have also used the COG and Pfam systems to examine how the functional diversity of a cluster varies along different levels of the HCG. About 40.8% of the clusters at the root level of the HCG each contain genes that have identical COG annotations, and 47.5% of the clusters at the root level each contain genes that have identical Pfam annotations. In contrast, these two percentages reach to 68.6% and 79.3% for the clusters at the middle level; and to 89.0% and 95.1% for the clusters at the leaf level. This indicates that from the root level down to the leaf level along the HCG hierarchy, the functional diversity of each cluster becomes increasingly less.

These validation results show that our clustering results are solid from both evolutionary and functional points of view.

4 Conclusion

We have described a framework for hierarchically classifying prokaryotic genes based on their functional equivalence relationships, which can be used to predict biological functions of genes at different levels and to provide hints on the evolutionary trace of genes/genomes. Besides, we have computationally validated the HCG by comparing the classification results for 224 prokaryotic genomes with the taxonomic hierarchy of these genomes, and with COG and Pfam. On one hand, these comparisons show that the HCG can essentially capture the information that is conveyable through these three systems. On the other hand, we have also shown that from the root levels down to the leaf levels along the HCG hierarchy the functional diversity of a cluster tends to be increasingly smaller, and the functional commonality shared by genes of the same cluster is increasingly more specific. We expect that this new functional classification scheme will provide a useful tool for gene predictions that is complementary to other classification schemes.

Acknowledgement. This work was supported in part by the US Department of Energy's Genomes to Life Program under project "Carbon Sequestration in *Synechococcus sp*: From Molecular Machines to Hierarchical Modeling" (http://www.genomes-to-life.org). The work is also supported, in part, by National Science Foundation (#NSF/DBI-0354771, #NSF/ITR-IIS-0407204, #NSF/DBI-0542119), and also by Distinguished Scholar grant from the Georgia Cancer Coalition.

References

[1] H. Su, J. Moniakis, and E. B. Newman, "Use of gene fusions of the structural gene sdaA to purify L-serine deaminase 1 from Escherichia coli K-12," *Eur J Biochem*, vol. 211, pp. 521-7, 1993.

[2] V. O. a. D. X. Ying Xu, "Clustering gene expression data using a graph-theoretic approach: An application of minimum spanning tree," *Bioinformatics*, vol. 18, pp. 526-535, 2002.

[3] V. Olman, D. Xu, and Y. Xu, "CUBIC: identification of regulatory binding sites through data clustering," *J Bioinform Comput Biol*, vol. 1, pp. 21-40, 2003.

[4] A. Kato and E. A. Groisman, "Connecting two-component regulatory systems by a protein that protects a response regulator from dephosphorylation by its cognate sensor," *Genes Dev*, vol. 18, pp. 2302-13, 2004.

[5] R. L. Tatusov, D. A. Natale, I. V. Garkavtsev, T. A. Tatusova, U. T. Shankavaram, B. S. Rao, B. Kiryutin, M. Y. Galperin, N. D. Fedorova, and E. V. Koonin, "The COG database: new developments in phylogenetic classification of proteins from complete genomes," *Nucleic Acids Res*, vol. 29, pp. 22-8, 2001.

[6] R. D. Finn, J. Mistry, B. Schuster-Bockler, S. Griffiths-Jones, V. Hollich, T. Lassmann, S. Moxon, M. Marshall, A. Khanna, R. Durbin, S. R. Eddy, E. L. Sonnhammer, and A. Bateman, "Pfam: clans, web tools and services," *Nucleic Acids Res*, vol. 34, pp. D247-51, 2006.

[7] H. Wu, F. Mao, V. Olman, and Y. Xu, "Hierarchical classification of functionally equivalent genes in prokaryotes," accepted by *Nucleic Acids Research*, 2007.

[8] S. F. Altschul, T. L. Madden, A. A. Schaffer, J. Zhang, Z. Zhang, W. Miller, and D. J. Lipman, "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs," *Nucleic Acids Res*, vol. 25, pp. 3389-402, 1997.

[9] A. Bairoch, "The ENZYME database in 2000," *Nucleic Acids Res*, vol. 28, pp. 304-5, 2000.

[10] P. Rice, I. Longden, and A. Bleasby, "EMBOSS: the European Molecular Biology Open Software Suite," *Trends Genet*, vol. 16, pp. 276-7, 2000.

[11] T. F. Smith and M. S. Waterman, "Identification of common molecular subsequences," *J Mol Biol*, vol. 147, pp. 195-7, 1981.

[12] M. D. Ermolaeva, O. White, and S. L. Salzberg, "Prediction of operons in microbial genomes," *Nucleic Acids Res*, vol. 29, pp. 1216-21, 2001.

[13] X. Chen, Z. Su, P. Dam, B. Palenik, Y. Xu, and T. Jiang, "Operon prediction by comparative genomics: an application to the Synechococcus sp. WH8102 genome," *Nucleic Acids Res*, vol. 32, pp. 2147-57, 2004.

[14] M. N. Price, K. H. Huang, E. J. Alm, and A. P. Arkin, "A novel method for accurate operon predictions in all sequenced prokaryotes," *Nucleic Acids Res*, vol. 33, pp. 880-92, 2005.

[15] Y. Zheng, J. D. Szustakowski, L. Fortnow, R. J. Roberts, and S. Kasif, "Computational identification of operons in microbial genomes," *Genome Res*, vol. 12, pp. 1221-30, 2002.

[16] U. M. F. a. K. B. Irani, "On the Handling of Continuous-Valued Attributes in Decision Tree Generation," *Machine Learning*, vol. 8, pp. 87-102, 1992.

[17] S. v. Dongen, "Graph Clustering by Flow Simulation," University of Utrecht, 2000.

[18] T. H. Cormen, *Introduction to algorithms*, 2nd ed. Cambridge, Mass.: MIT Press, 2001.

[19] R. C. Prim, "Shortest Connection Networks and Some Generalizations," *Bell System Technology Journal*, vol. 36, pp. 1389-1401, 1957.

[20] S. S. Wilks, *Mathematical Statistics*. New York: John Wiley & Sons, 1962.

[21] A. Balows, *The Prokaryotes : a handbook on the biology of bacteria : ecophysiology, isolation, identification, applications*. New York: Springer-Verlag, 1992.

[22] D. R. Boone, R. W. Castenholz, and G. M. Garrity, *Bergey's manual of systematic bacteriology*, 2nd ed. New York: Springer, 2001.

[23] E. Camon, M. Magrane, D. Barrell, D. Binns, W. Fleischmann, P. Kersey, N. Mulder, T. Oinn, J. Maslen, A. Cox, and R. Apweiler, "The Gene Ontology Annotation (GOA) project: implementation of GO in SWISS-PROT, TrEMBL, and InterPro," *Genome Res*, vol. 13, pp. 662-72, 2003.

[24] R. L. Tatusov, E. V. Koonin, and D. J. Lipman, "A genomic perspective on protein families," *Science*, vol. 278, pp. 631-7, 1997.

Using Multi Level Nearest Neighbor Classifiers for G-Protein Coupled Receptor Sub-families Prediction

Mudassir Fayyaz[1], Asifullah Khan[2], Adnan Mujahid[1], and Alex Kavokin[1]

[1] Faculty of Computer Science & Engineering,
Ghulam Ishaq Khan (GIK) Institute of Engineering Science & Technology, Swabi, Pakistan
mudassir.fayyaz@gmail.com
[2] Signal and Image Processing Lab, Deptt. of Mechatronics,
GIST, Gwangju, South Korea
asif_jg@yahoo.com

Abstract. Prediction based on the hydrophobicity of the protein yields potentially good classification rate as compared to the other compositions for G-Proteins coupled receptor (GPCR's) families and their respective sub-families. In the current study, we make use of the hydrophobicity of the proteins in order to obtain a fourier spectrum of the protein sequence, which is then used for classification purpose. The classification of 17 GPCR subfamilies is based on Nearest Neighbor (NN) method, which is employed at two levels. At level-1 classification, the GPCR super-family is recognized and at level-2, the respective sub-families for the predicted super-family are classified. As against Support Vector Machine (SVM), NN approach has shown better performance using both jackknife and independent data set testing. The results are formulated using three performance measures, the Mathew's Correlation Coefficient (MCC), overall accuracy (ACC) and reliability (R) on both training and independent data sets. Comparison of our results is carried out with the overall class accuracies obtained for super-families using existing technique. The multilevel classifier has shown promising performance and has achieved overall ACC and MCC of 97.02% and 0.95 using jackknife test, and 87.50 % and 0.85 for independent data set test respectively.

Keywords: Fast Fourier Transform, G-Proteins Coupled Receptors, Multilevel classification, Nearest Neighbor Classifier.

1 Introduction

The science of Bioinformatics, which is the melding of molecular biology with computer science, is essential to the use of genomic information in understanding human diseases and in the identification of new molecular targets for drug discovery [1].

The Protein bioinformatics aids in protein analysis and helps in suggestion and assigning of a function for all proteins that are known till present. Proteins are major constituent of living organisms and constitute more than 25% by weight of a cell [1]. They can perform variety of functions that involves digestion, transport, movement, sensory capabilities, immune protection and many more. G-protein coupled receptors (GPCR's) are transmembrane proteins that constitute largest protein family known. Its

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 564–576, 2007.

members are involved in all types of stimulus-response pathways, from intercellular communication to physiological senses. GPCR's consists of different amino acids sequences [2] and based on these, they are divided into six major classes. Most importantly, 50% of the drugs on the market today constitute GPCR's [3].

Amino acids and Nucleotides are the building blocks of proteins and act as intermediates in metabolism. There are 20 amino acids known today, found in a protein and convey a vast array of chemical versatility. The chemical properties of the amino acids of proteins specify its biological activity. The hydrophobic properties reflect the structure of protein.

Classifying GPCR's is an important job in the protein study. In past, various strategies have been employed for classification of GPCR's. These include similarity searching database search tools (e.g. BLAST, FASTA) [4-5] and such database searches coupled with searches of pattern databases (PRINTS) [6]. However, these methods fail when query proteins lack major sequence similarity to the database sequences. In order to overcome the limitations of the above two strategies, hidden markov model (HMM)-based methods [7-10] are used but these too have many limitations. Another approach of classifying GPCR's is based on SVM [11-14]. SVM is a binary classifier and approaches used in [11-14] need training *N* SVMs for *N* Protein families, which is a limitation of this approach. Further, to blend SVM for multi-classification, one need some weight selection mechanism or combination of different other models like HMM for combining the outputs from different SVMs. Some approaches have used weighted polling for combining the output of SVM, which again is an overhead.

The approach used in [11] is based on multi level classification of GPCR's using SVM. In this approach three level classification is performed. At first level the GPCR is validated, at level 2 the GPCR's super-families are predicated and at third level the GPCR's sub-families are predicted. This approach is good to follow as it increases the classification accuracy. The problem with this approach is that for S super-families and N sub-families predications, it needs to have 1+S+N classifiers, which is a big over head. On the other hand, the approach used in [12] uses 17 SVMs for classifying 17 sub-families belonging to different super families of proteins. It reports the results by performing jackknife test on the training data using one versus rest approach. The results on independent data set for only Class B sub-families are presented, while for the remaining sub-families, results are not reported, which raises doubts about the generalization performance of the proposed prediction model.

The approach proposed in the current work is based on classifying the 17 GPCR's sub-families based on 2-level NN classifier. In this way, in contrast to [11], only a total of 5 classifiers are utilized i.e. 1 classifier for level-1, GPCR's super-families predication and 4 classifiers for level-2, GPCR's sub-families predication (1 classifier for each super-family). This helps in developing fast, efficient and generalized prediction model for GPCR's sub-families.

2 Material and Methods

A. Data Set

In order to conduct our study, all the sequences belonging to subfamilies of Class B, C, D and F have been extracted from GPCRDB (March 2005, release 9.0) [36], by a text parsing method [21]. The sequences marked as putative/orphan are dropped out and identical sequences are removed except one. The subfamilies with less than 10 sequences are also dropped out because they are not going to play much role in classification. This process leads to 403 sequences, as in [13], belonging to 17 subfamilies. These sequences are then used as a training set for our classifier. As against [13], where independent data set performance of only class B has been mentioned, we extract the independent data set sequences for all the selected subfamilies of Class B, C, D and F from GPCRDB (June 2006 release 10.0). The sequence with putative/orphan and the sequence that have already been used for training are removed. This gives us a test set of 107 sequences belonging to the 17 subfamilies. The number of training and independent test sequences of each subfamily is listed in Table 1.

B. Protein Substitution

The three important properties of the protein, hydrophobicity, electronic property and bulk, are represented by hydrophobicity [18-20], electron-ion interaction potential (EIIP) [15, 16] and c-p-v [17] models respectively. These are used for converting the protein sequence to a numerical sequence. Hydrophobicity of proteins plays an important role in shaping a protein's structure and function. Three hydrophobicity scales, including KDHΦ [18], MHΦ [19] and FHΦ [20] have been mostly used by researcher for optimization. EIIP value describes the average energy states of all

Table 1. Number of Training & Test Sequences belonging to each G-Protein Coupled Receptor (GPCR) sub-families

Class	GPCR sub-family	Training Set	Test Set
Class B	Calcitonin	20	8
	Corticotropin releasing factor	23	6
	Glucagon	12	4
	Growth hormone-releasing hormone	13	3
	Parathyroid hormone	17	2
	PACAP	11	4
	Vasoactive intestinal polypeptide	14	3
	Latrophilin	20	10
	Methuselah-like proteins	21	4
Class C	Metabotropic glutamate	46	14
	Calcium-sensing like	18	6
	GABA-B	23	8
	Taste receptors	12	4
Class D	Fungal pheromone A-Factor like	16	5
	Fungal pheromone B like	32	7
Class F	Frizzled	94	17
	Smoothened	11	2
Total		403	107

valence electron of amino acids, while c-p-v model includes the composition (c), polarity (p), and molecular volume (v) of each amino acid. As proposed in [12], we have selected the best substitution model FHΦ in our experimentation for converting protein sequences to numerical sequences. The numerical series obtained after substitution is normalized to zero mean squared error by Equation (1).

$$x_{ij} = \frac{x_{ij} - \overline{x}_j}{s_j} \quad (1)$$

where x_{ij} is some property value of the *ith* amino acid residue in the *jth* sequence, $\overline{x}_j$ is the mean property value of the *jth* sequence, and s_j is the standard deviation of the *jth* sequence.

C. Power Spectral Density (PSD)

For classification purpose, we need all sequence to be of the same length. Proteins sequences that we have used are of variable lengths. The Fast Fourier Transform (FFT) has been frequently used in the protein bioinformatics for conversion of variable length sequence into a fixed length vector [21-23]. Based on the observations made in [12], we have used PSD of 512 frequency points for our experiments. The PSD is defined as a plot of power versus frequency. PSD serves as an input to the classifier. FFT is defined in Equation (2), while PSD is defined through Equation (3) and (4):

$$X(k) = \sum_{j=1}^{N} x(j)\omega_N^{(j-1)(k-1)} \quad (2)$$

$$CONJ(X) = REAL(X) - i * IMAG(X) \quad (3)$$

$$PSD = X \;.* \;\frac{CONJ(X)}{N} \quad (4)$$

where ω = exp (−2π*i* / N) is an *Nth* root of Unity. *N* denotes the frequency points and *CONJ* denotes complex conjugate.

D. The Nearest Neighbor Algorithm

In pattern recognition, the Nearest Neighbor algorithm (NN) is a method for classifying objects based on closest training examples in the feature space [26-28]. The training examples are mapped into multidimensional feature space and the space is partitioned into regions by class labels of the training samples. A point in the space is assigned to the class *C* if it is the closest class label among the training samples. Usually Euclidean distance metric is used to measure the proximity.

Let us assume that we have *N* proteins sequences ($\boldsymbol{X_1}, \boldsymbol{X_2} \ldots . \boldsymbol{X_N}$), each belonging to one of the *Y* classes (with labels $\mu_1, \mu_2 \ldots \mu_Y$). For a protein $\boldsymbol{X}$ under question, how

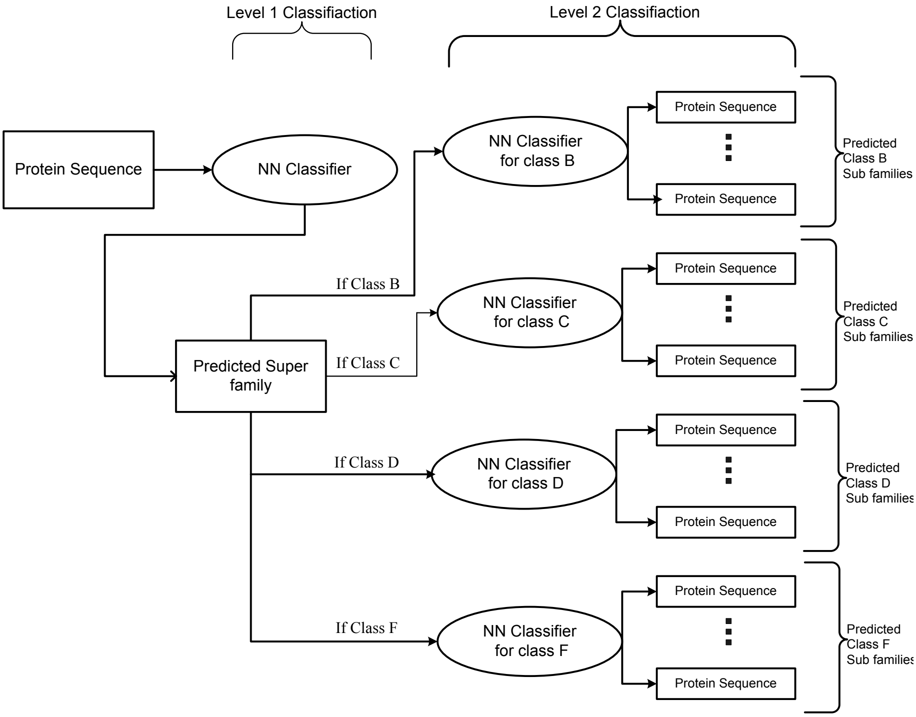

Level 1 Classifiaction
Level 2 Classifiaction
Protein Sequence
NN Classifier
Predicted Super family
If Class B
If Class C
If Class D
If Class F
NN Classifier for class B
NN Classifier for class C
NN Classifier for class D
NN Classifier for class F
Protein Sequence
Protein Sequence
Predicted Class B Sub families
Predicted Class C Sub families
Predicted Class D Sub families
Predicted Class F Sub families

Fig. 1. Flow Chart of the Proposed Scheme

can one predict its class label? According to the NN principle, we have to find the generalized distance between $\boldsymbol{X}$ and $\boldsymbol{X}_i$, as given in Equation (5).

$$\mathrm{D}(\boldsymbol{X}, \boldsymbol{X}_{\mathrm{i}}) = 1 - \frac{\boldsymbol{X} \cdot \boldsymbol{X}_{\mathrm{i}}}{\| \boldsymbol{X} \| \| \boldsymbol{X}_{\mathrm{i}} \|} \quad (i = 1, 2, \cdots, N) \tag{5}$$

where X · X_i is the dot product of vectors X and X_i, and ||X|| and $||X_i||$ are respectively their modulus.

The NN algorithm can be expressed as follows. First minimum generalized distance between X and X_k (k= 1, 2, . . . N) is computed as:

$$D(\boldsymbol{X}, \boldsymbol{X}_k) = Min\{D(\boldsymbol{X}, \boldsymbol{X}_1), D(\boldsymbol{X}, \boldsymbol{X}_2) \cdots, D(\boldsymbol{X}, \boldsymbol{X}_N)\} \tag{6}$$

The protein under question is assigned the category corresponding to the training protein $\boldsymbol{X}_k$.

E. Basic Methodology

The flow chart of the proposed scheme is given in Fig. 1. The more the number of classes are, the more it is possible that the classifier will confuse their class boundaries by overlapping some of them, hence affecting the over all accuracy.

However if we have less number of classes to be predicted, it is more probable that the classifier will construct the distinct boundary among the classes with a smaller overlapping margin. We have exploited this property in our work to achieve better classification accuracy. Classifying the super-families first at level-1 and based on level-1 predictions, classifying the respective sub-families at level-2 will reduce the classification inaccuracy as compared to the case where all subfamilies are predicted at a single level. The classification of sub-families in two levels will reduce the number of classes to be predicted by a single classier at any given time. In Table1, all the subfamilies are categorized into 4 super-families B, C, D and F containing 9, 4, 2 and 2 sub-families respectively.

F. Performance Evaluation

The independent data set, sub-sampling and jackknife tests are the three methods often used for cross-validation in statistical prediction. However, independent data set and Jackknife tests are important schemes of the performance evaluation [27-29]. In this study, we have used both independent data set and jackknife tests for cross validation of protein data. For the cross-validation by jackknifing, each of the proteins in the data set is in turn singled out as a test sample and remaining all samples are used to train the classifier. As in [12], four measures of performance are used for the evaluation of predictions by the classifier. They are ACC, MCC [30], total accuracy (ACC$_{\text{total}}$), total MCC (MCC$_{\text{total}}$), defined through Equations (7-18) respectively.

$$ACC_{LX}(i) = \frac{p(i)}{Z(i)} \tag{7}$$

where, *LX* represents the classification level for super-families and sub-families respectively; its possible values are 1 or 2, *i* range from 1 to the number of super-families (or sub-families). Z (*i*) is the number of samples in the super-family (or sub-family) *i*.

$$MCC_{LX}(i) = \frac{p(i)n(i) - u(i)o(i)}{\sqrt{[p(i)+u(i)][p(i)+o(i)][n(i)+u(i)][n(i)+o(i)]}} \tag{8}$$

p(*i*), n(i), o(i) and u(i) are the number of true positive, false negative, true negative and false positive sequences belonging to super-family (or sub-family) *i* .

$$ACC_{marginal}(k) = \frac{\sum_{j} p(j)}{N} \tag{9}$$

$$MCC_{marginal}(k) = \frac{\sum_{j} Z(j) MCC_{LX}(j)}{N} \tag{10}$$

where, *N* is the total number of protein sequences and *j* varies from 1 to the number of sub-families in a super-family *k*. $ACC_{marginal}$ and $MCC_{marginal}$ are the average ACC and MCC values for super-family *k* in level-2 classification respectively.

Table 2. Level-1 classification results based on the Hydrophobicity model (FHΦ) adopting 512 Frequency points, as validated by the Jackknife Test

Class	Proposed approach		
	ACC_{LX}	**MCC_{LX}**	**R**
Class B	98.01 %	0.97	0.98
Class C	99.0 %	0.99	0.99
Class D	97.92 %	0.95	0.99
Class F	98.10 %	0.97	0.99
Total	98.26%	0.98	

Table 3. Level-1 classification results based on the Hydrophobicity model (FHΦ) adopting 512 Frequency points, as validated by the Independent Data Test

Class	Proposed approach		
	ACC_{LX}	**MCC_{LX}**	**R**
Class B	95.45%	0.92	0.96
Class C	87.50 %	0.89	0.95
Class D	100.0%	0.92	0.96
Class F	100.0 %	0.97	0.97
Total	94.40%	0.92	

Table 4. Comparison of SVM and Multi Level classification results (Level-2) based on the Hydrophobicity model (FHΦ) adopting 512 Frequency points, as validated by the Jackknife Test

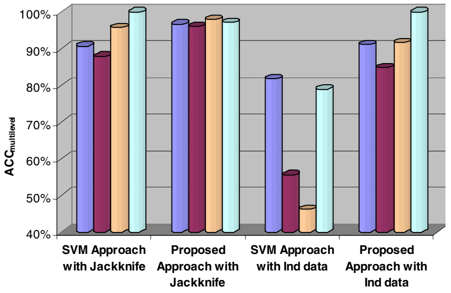
Class	SVM Based Method		Proposed approach	
	ACC	**MCC**	**ACC_{ML}**	**MCC_{ML}**
Class B	90.7%	0.94	96.71%	0.96
Class C	87.0%	0.91	96.03%	0.95
Class D	95.0%	0.97	97.92%	0.95
Class F	100%	1.0	97.16%	0.92
Overall	93.3%	0.95	96.80%	0.95

$$ACC_{total}(LX) = \frac{\sum_i p(i)}{N} \tag{11}$$

$$MCC_{total}(LX) = \frac{\sum_i Z(i) MCC_{LX}(i)}{N} \tag{12}$$

Table 5. Comparison of SVM and Multi Level classification results (Level-2) based on the Hydrophobicity model (FHΦ) adopting 512 Frequency points, as validated by the Independent Data Test

Class	SVM Based Method		Proposed approach	
	ACC$_{ML}$	**MCC$_{ML}$**	**ACC$_{ML}$**	**MCC$_{ML}$**
Class B	81.82%	0.90	91.11%	0.88
Class C	55.56%	0.77	84.77 %	0.85
Class D	46.15%	0.5	91.67%	0.77
Class F	78.95%	0.84	100.0 %	0.97
Overall	71.96%	0.81	90.86%	0.87



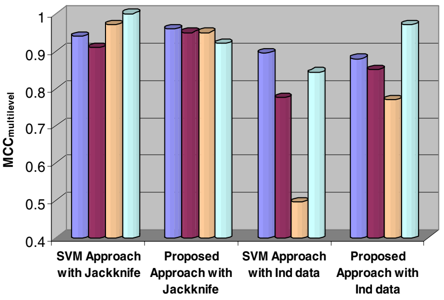


Fig. 2. Comparison of ACC$_{multilevel}$ of proposed scheme and ACC$_{total}$ of each superfamily in [12]

Fig. 3. Comparison of MCC$_{multilevel}$ of proposed scheme and ACC$_{total}$ of each superfamily in [12]

ACC$_{total}$ and MCC$_{total}$ represent the individual total ACC and MCC values for level-1 and level-2 classification steps for *LX*= 1 and 2 respectively.

$$ACC_{ML}(k) = 100 - [(100 - ACC_1(k)) + (ACC_1(k))\,(100 - ACC_{marginal}(k))] \tag{13}$$

$$MCC_{ML}(k) = 1 - [(1 - MCC_1(k)) + (MCC_1(k))\,(1 - MCC_{marginal}(k))] \tag{14}$$

ACC$_{ML}$ and MCC$_{ML}$ represent multilevel ACC and MCC value of each super-family, where ACC$_1$(*k*) and MCC$_1$(*k*) show the Accuracy and MCC value of *kth* superfamily at level-1.

$$ACC_{overall} = 100 - [(100 - ACC_{total}(1)) + (ACC_{total}(1))\,(100 - ACC_{total}(2))] \tag{15}$$

$$MCC_{overall} = 1 - [(1 - MCC_{total}(1)) + (MCC_{total}(1))\,(1 - MCC_{total}(2))] \tag{16}$$

ACC$_{overall}$ and MCC$_{overall}$ represent the multilevel overall ACC and MCC values for all super-families.

G. Reliability Prediction

When a machine learning approach is adopted, mostly the reliability prediction is an important factor. Equations (17, 18) define the reliability of prediction R [31] as:

$$R(i) = \frac{2*(ACC(i) - ERROR(i))}{1+ \mid ACC(i) - ERROR(i) \mid} \tag{17}$$

$$\text{where } ERROR(i) = \frac{o(i)}{n(i) + o(i)} \tag{18}$$

The reliability value lies between -1≤ R($_i$) ≤1. When all the receptors are correctly predicted; i.e. when Acc(*i*)=1 and Error(*i*)=0 then R(*i*)=1. On the other hand, R(*i*)=-1 when Acc(*i*)=0 and Error(*i*)=1.

3 Results and Discussions

A. Observations Using Jackknifing Test

FH scale of hydrophobicity has been used to convert protein sequences to numerical sequences, and further have been converted to a fixed length vector by FFT using 512 frequency points. In Table 2, columns 2-4 show the ACC, MCC and R values of each class (superfamily) after jackknife test in level-1 classification. The results show that ACC of each class is very promising and is approaching 100%. This consequently boosts the accuracy at level-2, which performs the actual prediction of GPCR sub-families. In Table 6, Columns 3-5 show the ACC, MCC and R respectively, belonging to 17 GPCR's sub-families for level-2 classification. The results show that the ACC_{LX} belonging to all subfamilies of class B is 100% for all of the subfamilies except Methuselah-like proteins, which is found to be 90.48. The results of the MCC_{LX} for subfamilies of class B also show significant results.Similarly the $ACC_{LX,}$ MCC_{LX} and R belonging to sub-families of class C, D and F improves too. The $ACC_{marginal}$ for class C, D and F are 96.97%, 100% and 99.05% respectively.

In Table 4, columns 2-3 show the ACC and MCC values of GPCR's classes in [12] and column 4-5 show the ACC_{ML} and MCC_{ML} obtained through proposed technique. From the results, it is clear that proposed technique outperforms the overall class results in [12] by a comprehensive margin for all the classes except class F, where it is lagging by 2.84%. The comparison of both techniques in terms of ACC and MCC for jackknife test is depicted in Figure 2 and 3 respectively. The result of R for each class and sub-families also shows the improvement. 0.4 better respectively than that of [12]. What we analyze from the experimental results is that sequential pattern based information using hydrophobicity could be exploited efficiently using an NN classifier that uses proximity in feature space for mapping feature space to class space.

Table 6. Level-2 classification results based on the Hydrophobicity model (FHΦ) adopting 512 even Frequency points, as validated by the Jackknife Test and Independent Data set Test

Class	GPCR sub-family	Proposed approach					
		Jackknife Test			Independent Data Test		
		ACC_{LX}	MCC_{LX}	R	ACC_{LX}	MCC_{LX}	R
Class B	Calcitonin	100%	1.0	1.0	100%	1.0	1.0
	Corticotropin releasing factor	100%	1.0	1.0	100%	1.0	1.0
	Glucagon	100%	0.96	1.0	100%	1.0	1.0
	Growth hormone-releasing hormone	100%	1.0	1.0	100%	1.0	1.0
	Parathyroid hormone	100%	1.0	1.0	100%	0.69	1.0
	PACAP	100%	1.0	1.0	100%	1.0	1.0
	Vasoactive intestinal polypeptide	100%	0.96	1.0	100%	1.0	1.0
	Latrophilin	100%	1.0	1.0	90%	0.93	0.99
	Methuselah-like proteins	90.48%	0.94	0.99	75.0%	0.85	0.98
	Marginal	98.68%	0.99		95.46%	0.96	
Class C	Metabotropic glutamate	100%	0.98	0.99	100%	1.0	1.0
	Calcium-sensing like	83.34%	0.90	0.96	100%	1.0	1.0
	GABA-B	100%	0.97	0.97	87.5%	0.92	0.98
	Taste receptors	100%	0.95	0.97	100%	0.88	0.97
	Marginal	96.97%	0.96		96.88%	0.96	
Class D	Fungal pheromone A-Factor like	100%	1.0	1.0	100%	0.84	0.92
	Fungal pheromone B like	100%	1.0	1.0	85.72%	0.84	0.96
	Marginal	100%	1.0		91.67%	0.84	
Class F	Frizzled	98.94%	0.95	1.0	100%	1.0	1.0
	Smoothened	100%	0.95	0.99	100%	1.0	1.0
	Marginal	99.05%	0.95		100%	1.0	
Total		98.51%	0.97		96.26%	0.95	

B. Results by Independent Test Data

The independent data set of 107 sequences involving the GPCR's from 17 sub-families has been validated by the classifiers trained on 403 sequences. In Table 3, columns 2-4 show the ACC, MCC and R values of each class (superfamily) after independent data set test in level-1 classification. The results show that ACC of class D and F is 100%, and for class B it is above 90% and 87% for class C. The scheme proposed in [12] has also been evaluated on independent data set. Our proposed scheme proves to go comprehensively ahead of SVM based method in [12], in terms of both ACC and MCC. In Table 6, columns 6-8 show the ACC and MCC and R respectively, belonging to 17 GPCR's sub-families for level-2 classification on independent data set. The results show that the ACC_{LX} belonging to all subfamilies of class B is 100% for all of the subfamilies except Latrophilin and Methuselah-like proteins, which are found to be 90% and 75% respectively. Similarly the ACC_{LX}, MCC_{LX}, and R belonging to sub-families of class C, D and F are better than that of [12].

In Table 5, columns 2-3 show the ACC and MCC values of GPCR's classes in [12] and column 4-5 show the ACC_{ML} and MCC_{ML} obtained through the proposed technique for independent data set. From the results, it is clear that proposed technique is better than that of [12], both in terms of ACC and MCC for all classes by a comprehensive margin. This comparison of both techniques in terms of ACC and MCC for independent data set test is depicted in Figure 2 and 3 respectively. The $ACC_{overall}$, and $MCC_{overall}$ are found to be 90.86% and 0.87 respectively. The results show that SVM based scheme in [12] fails to achieve generalization as compared to NN based proposed scheme. This is mainly because increase in margin of separation, which is the core concept of SVM classifiers, looses its effectiveness when used asone versus all strategy for blending it to multi-classification problem. The second reason could be that proximity, instead of increasing margin of separation, in the transform domain of protein sequences, has better discrimination power.

4 Conclusion

In this paper, a classification system using multi-level NN classifiers for the prediction of 17 GPCR's subfamilies is developed. Fourier spectrum of the amino acid composition of the sequences is used as the features for the NN classifiers. The results on both jackknife and independent data set test are better as compared to that of [12]. The reliability prediction has also been calculated for each sub-family and it improves too, which also validates our proposed technique. Based on the potential in classifying GPCR sub-families, in future, we intend to use ensembles of NN classifier using boosting [32] as well as Genetic programming [33, 34] based techniques.

Acknowledgements

We are very grateful to Ghulam Ishaq Khan Institute of Engineering Sciences and Technologies (GIKI) for providing healthy and rich research environment as well as moral support for carrying out this research. This work has been supported by National Engineering & Scientific Commission of Pakistan (NESCOM).

References

1. A. Tramantano: The 10 most wanted solutions in Bioinformatics, Champan & Hall/CRC Press (2005).
2. Eisenhaber,F. and Bork,P, Wanted: subcellular localization of proteins based on sequence, Trends Cell Biol (1998),vol. 8, pp. 69–70.
3. ilman, A.G., J.G. Hardman and L.E. Limbird: Goodman and Gilman's the pharmacological basis of therapeutics. McGraw Hill, New York.
4. Altschul SF, Gish W, Miller W, Myers EW, and Lipman DJ.: Basic local alignment search tool, J Mol Biol (1990), vol. 215, pp. 403–410.
5. Altschul SF, Madden TL, Schaffer AA, Zhang J, Zhang Z, Miller W, Lipman DJ. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs, Nucleic Acids (1997), vol. 25, pp. 3389–3402.
6. Lapinsh,M., Gutcaits,A., Prusis,P., Post,C., Lundstedt,T. and Wikberg,J.E.: Classification of G-protein coupled receptors by alignment-independent extraction of principal chemical properties of primary amino acid sequences, Protein Sci. (2002), vol. 11, pp. 795–805.

7. Karplus K, Barrett C and Hughey R.: Hidden Markov models for detecting remote protein homologie,. Bioinformatics (1998), vol. I4, pp. 846-856.
8. Sonnhammer E L L, Eddy S R, Birney E, Bateman A. and Durbin R., Pfam.: Multiple sequence alignments and hmm-profiles of protein domains (1998), NAR, vol. 26, pp. 320-322.
9. Papasaikas PK, Bagos PG, Litou ZI, Promponas VJ, Hamodrakas SJ.: PRED-GPCR: GPCR recognition and family classification server, Nucleic Acids (2004), vol. 32, pp. W380−W382.
10. Papasaikas PK, Bagos PG, Litou ZI, Hamodrakas SJ.: A novel method for GPCR recognition and family classification from sequence alone, using signatures derived from profile hidden Markov model., SAR QSAR Environ, vol. 14, pp. 413−420, 2003.
11. Bhasin M, Raghava GPS., GPCRpred: An SVM-based method for prediction of families and sub-families of G-protein coupled receptors, Nucleic Acids (2004), vol. 32: pp. W383−W389.
12. Guo YZ, Li ML, Wang KL, Wen ZN, Lu ML, Liu LX, Jiang L.: Fast Fourier transform-based support vector machine for prediction of G-protein coupled receptor sub-families, Acta Biochim Biophys Sin (2005), vol. 37, pp. 759–766.
13. Y.-Z. Guo, M. Li, M. Lu, Z. Wen, K. Wang, G. Li, and J. Wu.: Classifying G protein-coupled receptors and nuclear receptors on the basis of protein power spectrum from Fast Fourier transform, Amino Acids (2006), vol. 30, pp. 397–402.
14. Karchin R, Karplus K, Haussler D.: Classifying G-protein coupled receptors with support vector machines, Bioinformatics (2002), vol. 18, pp. 147−159.
15. Horn F, Vriend G, Cohen FE.: Collecting and harvesting biological data: The GPCRDB and NucleaRDB information systems, Nucleic Acids (2001), vol. 29, pp. 346−349.
16. Cosic I.: Macromolecular bioactivity: Is it resonant interaction between macromolecules? Theory and applications, IEEE Trans. Biomed (1994), vol. 41, pp. 1101−1114.
17. Grantham R.: Amino acid difference formula to help explain protein evolution, Science (1974), vol. 185, pp. 862−864.
18. Kyte J, Doolittle RF.: A simple method for displaying the hydropathic character of a protein, J Mol Biol (1982), vol. 157, pp. 105−132.
19. Mandell AJ, Selz KA, Shlesinger MF.: Wavelet transformation of protein hydrophobicity sequences suggests their memberships in structural families, Physica A (1997), vol. 244, pp. 254−262.
20. Fauchére J, Pliška V.: Hydrophobic parameters Φ of amino-acid side chains from the partitioning of n-acetyl-amino-acid amides, Eur J Med Chem Chim Ther (1983), vol. 18, pp. 369−375.
21. Hiramoto T, Nemoto W, Kikuchi T, Fujita N.: Construction of hypothetical three-dimensional structure of P2Y1 receptor based on Fourier transform analysis, J Protein Chem (2002), vol. 21, pp. 537–545.
22. Katoh K, Misawa K, Kuma K, Miyata T.: MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform, Nucleic Acids (2002), vol. 30, pp. 3059–3066.
23. Shepherd AJ, Gorse D, Thornton JM.: A novel approach to the recognition of protein architecture from sequence using Fourier analysis and neural networks, Proteins (2003), vol. 50, pp. 290–302.
24. T.M. Cover, P.E. Hart.: Inform. TheoryIT-13, IEEE Trans (1967), pp. 21–27.
25. J.H. Friedman, F. Baskett, L.J. Shustek.: Inform. TheoryC-24, IEEE Trans (1975), pp. 1000–1006.
26. R. O. Duda, P. E. Hart, D. G. Stork.: Pattern Classification, 2nd Edition, Wiley Press.

27. Chou KC, Zhang CT.: Prediction of protein structural classes, Crit Rev Biochem Mol (1995), vol. 30, pp. 275−349.
28. Chou KC, Cai YD.: Using functional domain composition and support vector machines for prediction of protein sub cellular location, J. Biol Chem (2002), vol. 277, pp. 45765–45769.
29. Chou KC, Cai YD.: Predicting protein structural class by functional domain composition, Biochem Biophys Res Commun (2005), vol. 321, pp. 1007–1009.
30. Matthews BW.: Comparison of predicted and observed secondary structure of T4 phage lysozyme, Biochim Biophys Acta (1975), vol. 405, pp. 442–451.
31. Novic M, Zupan J.: Investigation of infrared spectra-structure correlation using kohonen and counter-propagation neural network, J Chem Inf Comput Sci (1995), vol. 35, pp. 454–466.
32. Ludmila I. Kuncheva.: Combining Pattern Classifiers: Methods and Algorithms, John Wiley & Sons, Inc.2004
33. A. Khan, A. Majid, and Anwar M. Mirza.: Combination and Optimization of Classifiers in Gender Classification Using Genetic Programming, ISSN 1327-2314, International Journal of Knowledge-Based Intelligent Engineering Systems (2005), vol. 9, pp 1-11.
34. A. Majid, A. Khan and Anwar M. Mirza.: Combining Support Vector Machines Using Genetic Programming, International Journal of Hybrid Intelligent Systems (2006), vol. 3, No. 2, pp. 109-125.

Invited Talk: Ab Initio Gene Finding Engines: What Is Under the Hood

Mark Borodovsky

School of Biology, Department of Biomedical Engineering and College of Computing
Georgia Institute of Technology
Atlanta, Georgia, USA
mark.borodovsky@biology.gatech.edu

I will revisit the statistical and computational foundations of ab initio gene finding algorithms that best fit current challenges in analysis of genomic data. With the number of new sequenced genomes rapidly growing, there is a need to generate high quality gene annotations in less time.

In recent gene prediction competitions, the organizers described in great details the sets of experimentally confirmed eukaryotic genes that the contest participants were supposed to use for training statistical models, the key parts of ab initio gene finding algorithms. However, the gene prediction algorithm developed in our lab is only one of its kind that does not require a training set at all. It is using an unsupervised training approach and exhibits the same or better level of accuracy of gene identification as the algorithm trained on a sufficiently large training set. With more than 600 eukaryotic genome sequencing projects registered, as of February 2007, the self-learning gene finders become important tools able to accelerate extraction of biological information from newly sequenced eukaryotic genomes.

Another type of challenge in gene finding is presented by metagenomic sequences which are highly fragmented, diverse in nature, and carry larger rates of sequence irregularities than it is observed in sequenced genomes of cultivated microorganisms. The issues of finding gene starts or identifying short genes in metagenomes become much more difficult than in completely sequenced prokaryotic genomes.

Devising automatic gene annotation algorithms that identify specific features of gene organization in a novel genome and use adaptive strategies of self- training remains one of the open problems in machine learning. I will describe approaches to solving this problem for several classes of prokaryotic and eukaryotic genomes.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, p. 577, 2007.

Reconstruction of 3D Structures from Protein Contact Maps

Marco Vassura[1], Luciano Margara[1], Filippo Medri[1], Pietro di Lena[1], Piero Fariselli[2], and Rita Casadio[2]

[1] Computer Science Department
margara@cs.unibo.it
[2] Biocomputing Group, Department of Biology,
University of Bologna, Italy
casadio@alma.unibo.it

Abstract. Proteins are large organic compounds made of amino acids arranged in a linear chain (primary structure). Most proteins fold into unique three-dimensional (3D) structures called interchangeably tertiary, folded, or native structures. Discovering the tertiary structure of a protein (Protein Folding Problem) can provide important clues about how the protein performs its function and it is one of the most important problems in Bioinformatics. A contact map of a given protein *P* is a binary matrix *M* such that $M_{i,j} = 1$ iff the physical distance between amino acids *i* and *j* in the native structure is less than or equal to a pre-assigned threshold *t*. The contact map of each protein is a distinctive signature of its folded structure. Predicting the tertiary structure of a protein directly from its primary structure is a very complex and still unsolved problem. An alternative and probably more feasible approach is to predict the contact map of a protein from its primary structure and then to compute the tertiary structure starting from the predicted contact map. This last problem has been recently proven to be NP-Hard [6]. In this paper we give a heuristic method that is able to reconstruct in a few seconds a 3D model that exactly matches the target contact map. We wish to emphasize that our method computes an exact model for the protein independently of the contact map threshold. To our knowledge, our method outperforms all other techniques in the literature [5,10,17,19] both for the quality of the provided solutions and for the running times. Our experimental results are obtained on a non-redundant data set consisting of 1760 proteins which is by far the largest benchmark set used so far. Average running times range from 3 to 15 seconds depending on the contact map threshold and on the size of the protein. Repeated applications of our method (starting from randomly chosen distinct initial solutions) show that the same contact map may admit (depending on the threshold) quite different 3D models. Extensive experimental results show that contact map thresholds ranging from 10 to 18 Ångstrom allow to reconstruct 3D models that are very similar to the proteins native structure. Our Heuristic is freely available for testing on the web at the following url: http://vassura.web.cs.unibo.it/cmap23d/

1 Introduction

Protein folding is the process by which a protein assumes its three-dimensional (3D) structure. All protein molecules are heterogeneous chains of amino acids (or residues)

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 578–589, 2007.

which form the primary structure of the protein. Protein folding is in relation to the protein biological function. At the coarsest level, folding involves first the establishment of secondary structures, particularly alpha helices and beta sheets, and only afterwards tertiary structure. Actually, the greatest open problem in Structural Bioinformatics is the 3D protein structure prediction from its primary structure [13]. Since the protein folding problem is still unsolved, a typical alternative approach is to identify a set of sub-problems, such as the prediction of protein secondary structures, solvent accessibility and/or prediction of residue contacts and try to search specific solutions. Among different possibilities, the prediction of contact maps of proteins starting from the protein chain is particularly promising, since a partial solution of it can significantly help the prediction of the protein structure [9].

Having at hand a contact map a reliable and fast re-construction procedure of the 3D structure is needed. The problem is equivalent to unit disk graph realization which has been proved to be NP-hard [6]. Other well studied similar problems are NMR structure determination [12,16] and protein conformational freedom [11]. However the different nature of distance constraints induced by the contact map of the protein requires the use of other methods and tools. A series of heuristic algorithms have been developed to solve the problem. Galaktinov and Marshall [10] reconstructed the structures of five small proteins by adopting information relative to the residue coordination numbers. Vendruscolo et al. [19] described a method based on simulated annealing with the contact map as a target potential. They achieved an average RMSD of 2.5 Ångstrom (Å) on some 20 protein structures. Other approaches rely on steepest descent with inequality distance constraints [5] and on an algorithm which minimizes a continuous cost function that embodies constraints associated with contact and angle maps [17], respectively. On average these methods reconstruct the protein structures without completely satisfying the contact map in that the reconstructed protein structures may have contact maps that slightly differ from the native ones.

In this paper we propose a new heuristic algorithm for 3D reconstruction. We show that our algorithm is successful on our non redundant data set consisting of 1760 complete protein structures. Also, and irrespectively of the contact map threshold, it satisfies the initial contact map. Average execution times vary from 3 to 15 seconds, depending on the contact map threshold. The relation between contact map threshold, protein size and protein 3D structure is analyzed, showing that on average contact maps computed at thresholds between 10 and 18 Å allow a better 3D structure recovery than those computed at lower thresholds (from 7 to 9 Å).

2 Protein Structure Reconstruction

2.1 Protein Representation and Contact Map

Proteins structures are described by the coordinates of the atoms that concur to constitute the macromolecule. In this paper we adopt the widely used Cα representation of the protein backbone, where residues are considered as unique entities. The contact map of a given protein is a binary matrix *CM* such that $CM[i,j] = 1$ iff the Euclidean distance between residues *i* and *j* is less than or equal to a pre-assigned threshold *t*.

2.2 Distance Geometry and Protein Structure Reconstruction

Distance geometry (see [4] for an introduction) deals with the characterization of mathematical properties that can be derived from distance values between pairs of points. The mathematical foundation of distance geometry is essentially due to Cayley (1841) and Menger (1928) who showed how some basic geometry properties, such as convexity, could be defined in terms of distance values. One fundamental problem in distance geometry is to find a correct set of three-dimensional Euclidean coordinates that satisfy a set of distance constraints. In general, a set of points in the three-dimensional space that satisfy some given constraints does not exist. However, Cayley and Menger gave necessary and sufficient conditions for a set of positive values to be the exact distances between pairs of three-dimensional coordinates. Thus, given a consistent set of distances in the three-dimensional space the problem to find coordinates which satisfy such exact distance constraints can be solved by a polynomial-time algorithm [4] while the problem is NP-hard when the given set of distances is sparse [18].

NMR spectroscopy and X-ray crystallography are the most widely used experimental techniques to obtain bounds to the inter-atomic distances between residues. Because of experimental errors, we can usually obtain only a set of lower and upper bounds to such inter-atomic distances rather than exact values. The distance geometry-based approach to the protein structure reconstruction problem aims at developing techniques to recover the 3D protein structure, given a set of lower and upper bounds to residue inter-atomic distances. The problem of computing a set of consistent coordinates is generally intractable [15]. Havel and Crippen developed a recovering algorithm from a sparse set of lower and upper bounds to the inter-atomic distances [8,12]. Their algorithm first uses some bound smoothing technique to estimate bounds values for the missing distances. Then it uses an algebraic technique known as the EMBED algorithm to generate an approximate set of three-dimensional coordinates adopted as starting solution for an optimization procedure. While the problem of recovering protein structures from a set of distances is known to have a polynomial time solution, the same problem from contact maps is NP-hard [6]. However, empirical developed applications seem to suggest that such approach is fruitful (see for example [5, 19, and 20]). An introduction to the approach of predicting protein structure from contact maps can be found elsewhere [3].

3 Algorithm Description

In this section we briefly describe a heuristic algorithm which finds a set of three-dimensional coordinates consistent with some given contact map *CM* of threshold *t*.

The algorithm is in two phases (see pseudocode below): in the first phase it generates a random initial set of 3D coordinates $C \in \boldsymbol{R}^{3\times n}$ while in the second phase it refines the set of coordinates by applying a correction/perturbation procedure. The refinement applies until the set of coordinates is consistent with the given contact map or until a control parameter $\mathcal{E}$ becomes 0. The control parameter $\mathcal{E}$ has initially a positive value and it is decremented every some amount of refinement steps. If it reaches the 0 value before a consistent set of coordinates is found, then a new random initial set of coordinates is generated; $\mathcal{E}$ is initialized again to a strictly positive value and the refinement procedure re-starts from the beginning.

```
HEURISTIC RECONSTRUCTION($CM \in \{0,1\}^{n \times n}$, $t \in N$)
while coordinates set C is not correct do
    //First phase: random generation
    $C \leftarrow$ RANDOM-PREDICT($CM, t$)
    //Second phase: refinement
    $C \leftarrow$ CORRECT($CM, C, t$)
    set $\mathcal{E}$ to a strictly positive value
    while coordinates set C is not consistent with CM and $\mathcal{E} > 0$ do
        $C \leftarrow$ PERTURBATE($CM, C, t, \mathcal{E}$)
        $C \leftarrow$ CORRECT($CM, C, t$)
        decrement slightly $\mathcal{E}$
return C
```

The first phase of the algorithm consists in the partially random prediction (**RANDOM-PREDICT**) of a set of starting coordinates (as consistent as possible with a given contact map) that will be the starting point for the refinement procedure in the second phase of the algorithm. A fast and reliable way to obtain good starting coordinates is provided by the *metric matrix embedding* (EMBED) algorithm [12]. Generally, no set of three-dimensional points is consistent with some distance matrix D. However, the EMBED algorithm can be used to compute a set of three-dimensional coordinates that is, in a certain sense, the best three-dimensional fit for D. The computing of the initial solution is preceded by a phase in which the contact map is scanned for the existence of *splittable* components. Splitting the initial contact map in submatrices is done to locate those fragments of proteins which demonstrate a high degree of independence with respect to mutual interactions. The submatrices are then separately used to create sets of coordinates to be merged in order to give an initial solution. The merging procedure is managed by selecting, between a set of equally distributed three-dimensional angles, the best rotation with respect to the lower number of errors generated in the contact map. A more detailed description of the prediction phase can be found in [14].

The second phase of the algorithm iteratively applies two local techniques to the current set of coordinates, **CORRECT** and **PERTURBATE**. This is performed in order obtain a new set of coordinates “more consistent” with the given contact map. We call *not well placed* residues whose coordinates are not consistent (according the contact map) with the coordinates of all other residues. The local correction technique **CORRECT** attempts to change the coordinates of every not well placed residue i in order to reduce the cardinality of the set of not well placed residues. The *radius of mobility* r_i of the residue i is defined as $r_i = \min\{D_0 - t, t - D_1\}$ where $D_0 = \min\{d_{ij}|d_{ij} > t$ and $CM[i,j] = 0\}$ and $D_1 = \min\{d_{ij}|d_{ij} \leq t$ and $CM[i,j] = 1\}$. Then, the residue i can safely move to the surface of the sphere with center i and radius r_i without decreasing, and eventually increasing, the cardinality of the set of residues well placed. The new position has to be as distant as possible from the whole set of residues j not well placed wrt i such that $CM[i,j] = 0$ and as close as possible to the whole set of residues *k* not well placed wrt *i* such that *CM*[*i,k*] = 1. This is achieved by projecting the *C*[*i*] coordinates on the surface of the sphere in the direction described by a force vector *F* applied to *i* (line 7). For every residue *j* not well placed wrt *i*, let consider the (vectorial) pseudo-force $F_j = (C[i] - C[j])/\|C[i] - C[j]\|$ of magnitude one and direction *ij*. *F* is the result of the (vectorial) addition of all F'_j, where $F'_j = F_j$ whether *CM*[*i,j*] = 1 and $F'_j = -F_j$ has opposite direction to F_j whether *CM*[*i,j*] = 0 (lines 5-6).

```
CORRECT($CM \in \{0,1\}^{n \times n}$, $C \in \mathbf{R}^{3 \times n}$, $t \in \mathbf{N}$)
1: foreach $i \in \{1,...,n\}$ not well placed do
2:     $F \leftarrow \{0, 0, 0\}$
3:     foreach $j \in \{1,...,n\}$ not well placed wrt i do
4:             compute the radius of mobility $r_i$ according to CM and C
5:             if $CM[i,j] = 1$ then $F \leftarrow F - \frac{C[i]-C[j]}{\|C[i]-C[j]\|}$
6:                         else $F \leftarrow F + \frac{C[i]-C[j]}{\|C[i]-C[j]\|}$
7:     $C[i] \leftarrow C[i] + F \cdot \frac{r_i}{\|F\|}$
8: return C
```

A run of the correction procedure does not add new errors to the coordinates set, but may reduce the radius of mobility for not well placed residues. In order to maintain as large as possible the radius of mobility for such residues, after a correction procedure we apply small perturbations to the coordinates set using the **PERTURBATE** procedure. For every residue *i* and every residue *j* well placed wrt *i*, if the distance d_{ij} is under a given threshold then **PERTURBATE** changes the coordinates of *i* and *j* in order to make them a bit more closer (lines 3-4) and if d_{ij} is above a given threshold then **PERTURBATE** changes the coordinates of *i* and *j* in order to make them a bit more distant (lines 5-6). A perturbation can introduce new errors to the coordinates set but, conversely, it avoids that not well placed residues get stuck.

```
PERTURBATE($CM \in \{0,1\}^{n \times n}$, $C \in \mathbf{R}^{3 \times n}$, $t \in \mathbf{N}$, $\varepsilon \in \mathbf{R}$)
1: for $i \leftarrow 1$ to $n$ do
2:     for $j \leftarrow 1$ to $n$ do
3:             if $t - \varepsilon < \|C[i] - C[j]\| \le t$ and $CM[i,j] = 1$ then
4:                     approach $C[i]$ and $C[j]$ of $\varepsilon/10$
5:             if $t < \|C[i] - C[j]\| < t + \varepsilon$ and $CM[i,j] = 0$ then
6:                     place at distance $C[i]$ and $C[j]$ of $\varepsilon/10$
7: return C
```

4 Experimental Results

We selected the list of proteins with their relative structural classifications from SCOP [2] release 1.67. We then downloaded the corresponding protein structures from the PDB and we retained only those files with coordinates obtained with X-ray experiments, with resolution <2.5 Å, and without missed internal residues. Finally, using BLAST [1] we removed sequence redundancies, ending up with a datasets of 1760 protein chains with sequence similarity lower than 25%. The distribution of the 1760 protein chains accordingly to the SCOP classification is shown in Fig. 1. Our protein set contains 1502 one domain- and 258 multi-domain chains. The complete list is available at the web site http://vassura.web.cs.unibo.it/protlist.tgz.

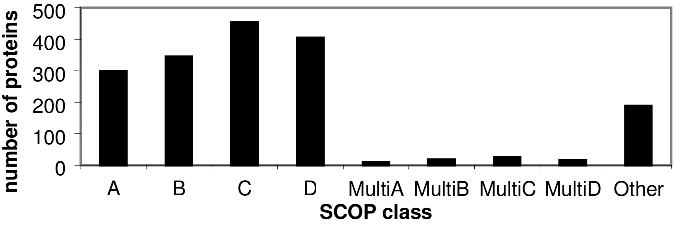


Fig. 1. Distribution of our protein set according to the SCOP classes. A=all alpha; B=all beta; C=Alpha/Beta; D=Alpha+Beta. Multi-{A,B,C,D} and Other contain multi-domain proteins.

All the test runs are executed on personal computers equipped with an Intel Pentium 4 processor with a clock rate of 3GHz and 1Gb of RAM memory. Times reported are Unix user CPU times, and are measured using the `time()` C library function. During each run the program collects time information before reading the input and again after computing the result; the CPU time actually elapsed is computed as the difference between the two figures. The Heuristic is freely available for testing on the web at the following url: http://vassura.web.cs.unibo.it/cmap23d/

To measure the difference between contact maps, we used the simple Hamming distance that counts the numbers of different bits; this distance is also the target function of the problem. When we deal with two protein structures, the classical Root Mean Square Deviation (RMSD) is computed between the native and the reconstructed structure. RMSD is commonly used to compare two molecular structures described by some set of coordinates C, $C' \in \boldsymbol{R}^{3\times n}$.It is defined as the smallest distance

$$D_k = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(C'[i] - C_k[i])^2}$$

where $C_k \in \boldsymbol{R}^{3\times n}$ is obtained by rotating and translating the coordinates set C.

For each protein of our selected non redundant data set, containing 1760 protein structures (see Protein set section), we generate 12 different contact maps by changing the contact threshold from 7 to 18 Å with a 1 Å step and then we run our procedure for all the 12*1760 generated contact maps.

The most relevant result of our procedure is the fact that all the reconstructed protein structures satisfy the native contact maps. This means that the Hamming distance between the native and the reconstructed contact maps is 0, or in other words, that given the contact map of a protein our algorithm finds a 3D structure which has the same contact map of the native protein. In spite of this, in some cases, the RMSD of the reconstructed protein with respect to the native structure can be very large (Fig.2).

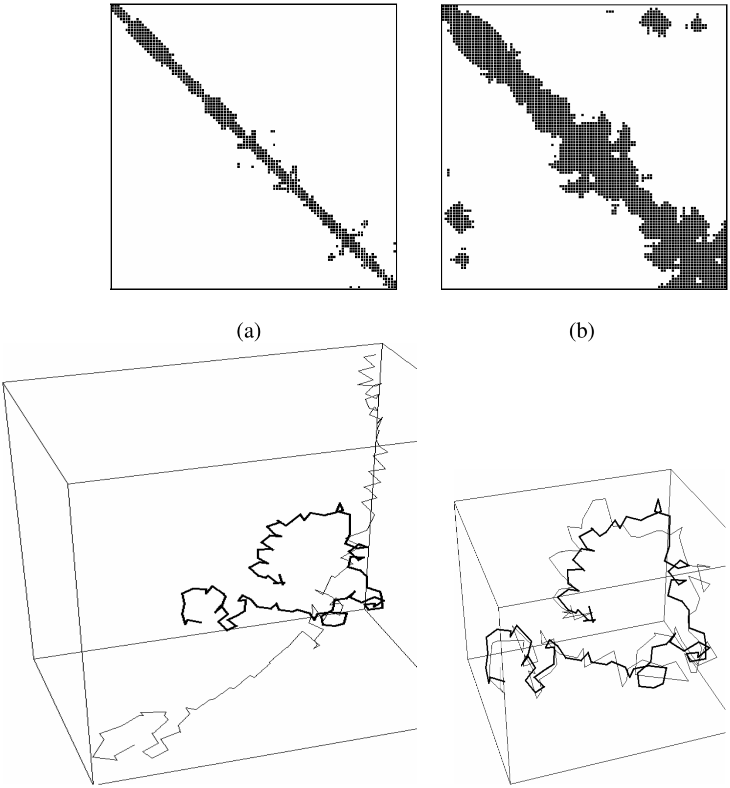


Fig. 2. Contact map degeneracy: a test case. The recovery of the 3D structure of 1cxp chain B (104 residues, all-alpha). (a) 1cpx contact map computed at a threshold of 7 Å; (b) 1cpx contact map computed at a threshold of 16 Å; (c) 1cpx native structure (thick line) compared to a recovered structure with the same contact map (a) (RMSD= 41.31Å); (d) 1cpx native structure (thick line) compared to a recovered structure with the same contact map (b) (RMSD= 4.95Å).

This indicates that some contact maps can represent a huge ensemble of protein conformations and degenerate. Usually this means that the map contains only a broad central band of local contacts, and no constraints are posed on the global bending of the protein. The reconstruction ambiguity is more evident when the contact map is generated using low values of contact thresholds (ranging from 7 to 9 Å) and decreases as the contact threshold increases (Table 1). Our results indicate that at increasing contact map threshold both average RMSD and standard deviation values decreases over the all protein set (Table 1). At increasing threshold value global features in the contact map help in finding the 3D structure likely to be more similar/close to the native one.

Table 1. Scoring the recovery of 3D structure from the contact maps of 1760 proteins

Threshold (Å)	Cmap dist (Å)	Avg RMSD (Å)	AvgSD RMSD (Å)	Avg Time (s)	AvgSD Time (s)
7	0	6.11	4.09	15	136
8	0	4.58	3.86	9	110
9	0	3.37	3.42	9	155
10	0	2.62	2.98	10	157
11	0	2.21	2.69	5	71
12	0	1.97	2.51	3	15
13	0	1.75	2.29	2	13
14	0	1.58	2.09	3	16
15	0	1.47	2.01	10	274
16	0	1.39	1.90	2	9
17	0	1.36	1.75	5	94
18	0	1.35	1.79	3	17

Threshold is the threshold used to compute the input contact map; *Cmap dist* is the Hamming distance between the contact map of the native structure and the contact map of the recovered structure; *Avg RMSD* is the average, over all proteins, RMSD between the native structure and the recovered structure; *AvgSD* is the average standard deviation over all proteins; *Avg Time* is the average, over all proteins, time needed to recover the 3D structure.

A typical example is shown in Fig. 2 for protein 1cxp chain B. The contact map computed with a threshold equal to 7 Å (Fig. 2a) does not contain enough global information of the protein structure and a large number of protein structures is represented by that map. For instance, a possible reconstruction is reported in Fig. 2c where the RMSD to the native structure is 41.3 Å. When the contact map is computed at a threshold of 16 Å (Fig. 2b) more features appear off of the main diagonal and the recovered 3D structure is closer to the native one. Indeed RMSD decreases now to 4.9 Å (Fig. 2d).

This finding prompted us to do a search in the threshold space to optimize the RMSD values. We find that a better 3D reconstruction is obtained when a high threshold value is adopted (10 Å or higher), while the average running time (over 1760 proteins) does not depend on the threshold adopted (Table 1). RMSD values between the reconstructed and the corresponding native 3D protein structures are analyzed as function of the four main SCOP classes, clustered in mono and multi-domain proteins. The results are shown in Fig. 3. As a general trend we find that multi-domain proteins are more easily reconstructed with our procedure than mono domain proteins. This is so rather independently of the threshold value adopted. One possible explanation is that the contact map of multi-domain proteins carry information about the inter-domain residue contacts that poses more constraints to the reconstruction of the 3D protein structure. Another interesting point that emerges from Fig. 3 is the fact that the contact maps of mono-domain all-alpha proteins (A SCOP label) tend, on average, to be more ambiguous in their reconstruction. This is in agreement with the fact that all-alpha proteins are characterized by contact maps with a great number of contacts made by sequence nearest-neighbor residues and this hampers global 3D reconstruction.

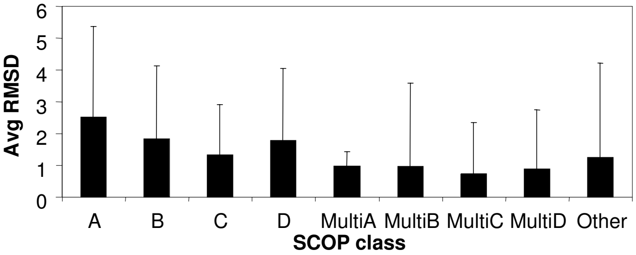


Fig. 3. Average RMSD on the different SCOP classes, obtained using contact maps computed with a threshold of 13 Å

An analysis of our procedure as function of the protein length, show that the method is working independently of the protein size and that long proteins are on average reconstructed as well as short ones (Fig. 4).

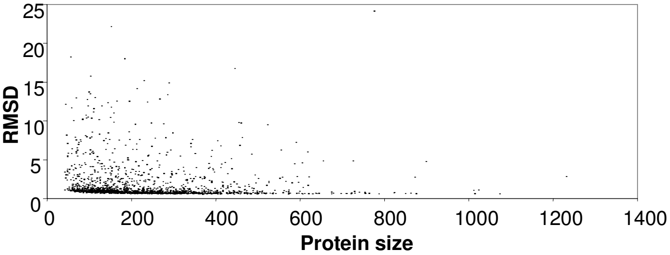


Fig. 4. Actual RMSD distribution as function of the protein length when contact maps are computed with a contact threshold of 13 Å

4.4 Comparison with Previous Methods

To our knowledge only four methods have been introduced so far to reconstruct the protein 3D structures starting from the contact map information [5,10,17,19]. The approach developed by Vendruscolo et al. [19] was tested on some 20 proteins. Different from our results, their finding indicates that RMSD on average increases when the protein length increases.

This effect may be due to the adopted simulated annealing procedure that require more optimization steps for large than for short proteins; furthermore they stop the search without a complete satisfaction of the contact maps (Cmap distance = 0). On the contrary, our method runs till the very satisfaction of the contact map (Table 1).

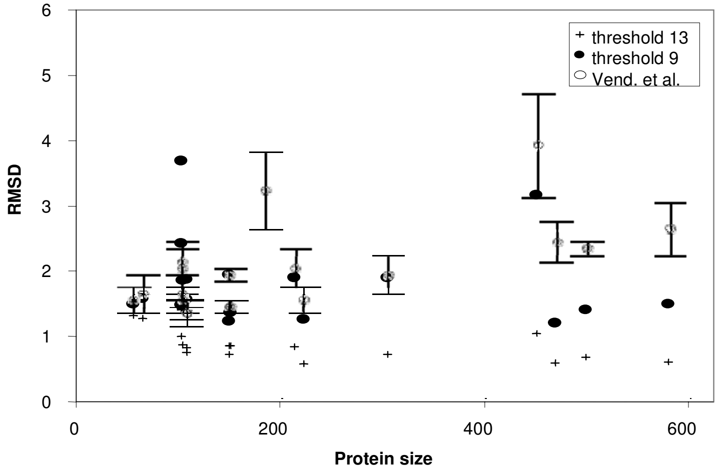


Fig. 5. Reconstruction accuracy (RMSD) of our method on the set of Vendruscolo et al, [19]. The results correspond to a contact threshold of 9 Å (for a direct comparison with [19]) and of 13 Å, respectively. The error associated with the Vendruscolo et al. reported data is due to the fact that the complete satisfaction of the contact map is not a constraint for their search.

Table 2. Comparison of our method with that of Galaktinov and Marshall [10]

Protein	**Number of residues**	**Galaktinov Marshall [10] RMSD (Å)**	**Our method(*) RMSD (Å)**	**Time (s)**
1rdg	52	0.66	1.08	0.01
1pcy	99	0.88	0.90	0.08
4fd1	106	0.86	0.74	0.14
1acx	108	0.96	0.83	0.13
1cpv	108	0.89	0.80	0.12
Average		0.85	0.87	0.096

The protein set is the same of Galaktinov and Marshall [10].

(*) In this specific case we used a cut-off threshold of 13 Å; the results with other thresholds are similar.

When our method is tested on the Vendruscolo et al set [19], it is worth noticing that even when a comparable threshold of 9 Å is used, the reconstructed RMSD is lower on average then that previously obtained (Fig. 5). At higher contact map threshold value (for instance 13 Å, as shown in Fig. 5), all the proteins of the Vendruscolo et al. set [19] are reconstructed with RMSD values lower than 2 Å and again with 0 errors in the contact map. The average execution time on this set is less than 1 second.

Galaktinov and Marshall [10] reports values for only for 5 proteins, with RMSD values lower than 1 Å. In Table 2 we show that our method performs similarly on their data set (results were obtained with a contact threshold of 13 Ås).

Two other papers [5, 17] describe reconstruction procedures: however they adopt predicted constraints or predicted contacts to fold the proteins, so that a direct comparison with them is not possible.

5 Conclusions and Further Works

In this paper we address the problem of reconstructing the protein structures starting from the contact matrix. We introduce this problem and we describe an efficient and very fast procedure. We show that contact maps computed using threshold values greater than those commonly used for Cα-Cα distances allow better 3D structure recovery than those computed at lower thresholds (7-9 Å). This is mainly due to the fact that for some proteins (in particular, but not exclusively, the all-alpha mono-domain) exist a large number of different conformations that satisfy the native contact map. When the threshold of the contact map computation is increased the ensemble of possible different solutions is reduced by increasing the number of structural constraints. Our finding indicates that the best cut-off threshold is in the range of 10-18 Å.

In summary in this paper we show that:

- our method can reconstruct with zero contact map errors all the protein structures of our data set, and to our knowledge, this result has not been achieved before by other authors;
- the required computational time is in the range of few seconds when a normal personal computer is available, making the program a useful tool also for wide-scale applications;
- our results are obtained on a non-redundant data set comprising 1760 proteins and this is the largest dataset used so far for this specific task.

We are working on reconstruction of protein 3D structure from noisy, and possibly non-physical, contact maps. Obviously it is not always possible to reconstruct with zero contact map errors, nevertheless, comparing reconstructed structure with the native one, some preliminary tests show that our method is tolerant and reliable.

References

1. S.F. Altschul, T.L. Madden, A.A. Schaffer, J. Zhang, Z. Zhang, W. Miller, D._J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Res. 1997 Sep 1;25(17):3389-402
2. Andreeva, D. Howorth, S.E. Brenner, T.J. Hubbard, C. Chothia, A.G. Murzin . SCOP database in 2004: refinements integrate structure and sequence family data. Nucleic Acids Res. 2004 Jan 1;32(Database issue):D226-9
3. L. Bartoli, E. Capriotti, P. Fariselli, P.L. Martelli, R. Casadio. The pros and cons of predicting protein contact maps.
4. L.M. Blumental. Theory and applications of distance geometry, Chelsea, New York 1970.
5. J. Bohr, et al. Protein structures from distance inequalities. J. Mol. Biol. 231, 861-869, 1993.
6. H. Breu, D.G. Kirkpatrick, *Unit disk graph recognition is NP-hard*, Computational Geometry 9 (1998) 3-24.
7. T. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. Introduction to algorithms. Second edition. *MIT Press, Cambridge, MA; McGraw-Hill Book Co., Boston, MA,* 2001.
8. G.M. Crippen, T.F. Havel. Distance geometry and molecular conformation. John Wiley & Sons, 1988.

9. P. Fariselli, O. Olmea, A. Valencia, R. Casadio. Progress in predicting inter- residue contacts of proteins with neural networks and correlated mutations. Proteins: 45 Suppl 5: 157-162 (2001)
10. S.G. Galaktionov, G.R. Marshall. Properties of intraglobular contacts in proteins: an approach to prediction of tertiary structure. In System Sciences, 1994. Vol.V:, Proceedings of the Twenty-Seventh Hawaii International Conference on Biotechnology Computing Vol. 5, 4-7 Jan. 1994 Page(s):326 – 335
11. B.L. de Groot, D.M.F. van Aalten, R.M. Scheek, A. Amadei, G. Vriend and H.J.C. Berendsen; Prediction of protein conformational freedom from distance constraints, Proteins 29: 240-251, 1997
12. T.F. Havel. Distance Geometry: Theory, Algorithms, and Chemical Applications in the Encyclopedia of Computational Chemistry 1998.
13. A. Lesk. Introduction to Bioinformatics. Oxford University Press, 2006.
14. L. Margara, M. Vassura, P. di Lena, F. Medri, P. Fariselli, R. Casadio. Reconstruction of the Protein Structures from Contact Maps. University of Bologna, Department of Computer Science, technical report UBLCS-2006-24, October 2006.
15. J. Moré, Z. Wu. [epsilon]-Optimal solutions to distance geometry problems via global continuation. In P. M. Pardalos, D. Shalloway, and G. Xue, editors, Global Minimization of Nonconvex Energy Functions: Molecular Conformation and Protein Folding, pages 151–168. American Mathemtical Society, 1995.
16. J. Moré, Z. Wu. Distance geometry optimization for protein structures**,** Journal on Global Optimization, 15, pp. 219-234, 1999
17. G. Pollastri, A. Vullo, P. Fiasconi, P. Baldi. Modular DAG-RNN Architectures for Assembling Coarse Protein Structures J. Comp. Biol., 13:3, 631-650, 2006
18. J. B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. In Proc. 17th Allerton Conf. Commun. Control Comput., pages 480–489, 1979.
19. M. Vendruscolo, E. Kussell, and E. Domany. Recovery of protein structure from contact maps. Folding and Design, 2(5):295? 306, September 1997.
20. M. Vendruscolo, E. Domany. Protein folding using contact maps. Vitam Horm 2000, 58, 171-212.

A Feature Selection Algorithm Based on Graph Theory and Random Forests for Protein Secondary Structure Prediction

Gulsah Altun[1], Hae-Jin Hu[1], Stefan Gremalschi[1], Robert W. Harrison[1,2], and Yi Pan[1]

[1] Department of Computer Science
[2] Department of Biology
Georgia State University
30303, Atlanta, GA, USA

Abstract. Protein secondary structure prediction problem is one of the widely studied problems in bioinformatics. Predicting the secondary structure of a protein is an important step for determining its tertiary structure and thus its function. This paper explores the protein secondary structure problem using a novel feature selection algorithm combined with a machine learning approach based on random forests. For feature reduction, we propose an algorithm that uses a graph theoretical approach which finds cliques in the non-position specific evolutionary profiles of proteins obtained from BLOSUM62. Then, the features selected by this algorithm are used for condensing the position specific evolutionary information obtained from PSI-BLAST. Our results show that we are able to save significant amount of space and time and still achieve high accuracy results even when the features of the data are 25% reduced.

Keywords: protein secondary structure prediction, feature selection, random forests, clique.

1 Introduction

Proteins play a variety of roles that define particular functions of a cell. They interact between DNA, RNA and other proteins in their tertiary and quaternary state. Therefore, knowing the structure of a protein is crucial for understanding its function. A protein is primarily made up of amino acids which determine its structure. The structure of a protein exists in three levels. The first level is the secondary structure which is formed of recurring shapes called alpha-helix, the beta-sheet, and coil. The tertiary structure of a protein is the spatial assembly of helices and sheets and the pattern of interactions between them. Many proteins contain more than one subunit and the combinations of these subunits are called the quaternary structure.

Today, large volumes of genes are being sequenced. Therefore, the gap between known protein sequences and protein structures being solved by experimentally is growing exponentially. Today, in PDB there are over 1 million proteins whose amino acid sequence are known, however only about 40,000 of these proteins' structures are known [3][4]. The reason for this gap is that NMR and x-cryptography techniques take

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 590–600, 2007.

years to determine the structure of one protein. Therefore, having computational tools to predict the structure of a protein is very important and necessary. Even though most of the computational methods proposed for protein structure prediction do not give 100% accurate results, even an approximate model can help experimental biologists for guiding their experiments.

Predicting the secondary and tertiary structure of a protein from its amino acid sequence is one of the important problems in bioinformatics. However, with the methods available today, protein tertiary structure prediction is a very hard task even when starting from the exact knowledge of protein backbone torsion angles [11]. It is also suggested that protein secondary structure delimits overall topology of the proteins [23]. It is believed that predicting the protein secondary structure gives an insight and an important starting point for the prediction of the tertiary structure of the protein, which leads to understanding the function of the protein.

The organization of this paper is as follows: In section 2, we present the problem formulation and previous work on the protein secondary structure prediction problem. We also present a brief background on the random forests and some feature selection techniques used for protein secondary structure prediction. In section 3, we propose a novel algorithm using a graph theory approach for feature selection based on evolutionary information stored by the BLOSUM62 matrix [14]. We use this method to select features from PSSM (Position Specific Scoring Matrix) profiles of proteins [1]. In section 4, we show our experimental results and compare different encoding schemes of proteins. In section 5, we give conclusion and future work.

2 Problem Formulation and Background

2.1 Protein Secondary Structure Prediction Problem Formulation

This study adopted the most generally used DSSP secondary structure assignment scheme. The DSSP classifies the secondary structure into eight different classes: H (α- helix), G (3_{10}-helix), I (π-helix), E (β-strand), B (isolated β-bridge), T (turn), S (bend), and - (rest). These eight classes were reduced into three regular classes based on the following method: H, G and I to H; E to E; all others to C.

The problem formulation is stated as: Given a protein sequence $a_1a_2 \ldots a_N$, secondary structure prediction is finding the state of each amino acid a_i as being either H (helix), E (beta strand), or C (coil). The quality of secondary structure prediction is measured with a "3-state accuracy" score called Q_3. Q_3 is the percent of residues that match reality. Most of the previous research adopted Q_3 as an accuracy measurement.

2.2 Previous Work on Protein Secondary Structure Prediction

Protein secondary structure prediction problem has been studied widely for almost a quarter of a century. Many methods have been developed for prediction of the secondary structure of proteins. In the initial approaches, secondary structure predictions were performed on single sequences rather than families of homologous sequences [12]. The methods were shown to be around 65% accurate. Later, with the availability of large families of homologous sequence, it is found out that when these methods were applied to a family of proteins rather than a single sequence, the accuracy increased well above 70%. Today, many proposed methods utilize evolutionary information such as multiple alignments and PSI-BLAST profiles [1] [13]. Many of these methods that are based on Neural networks, SVM and hidden markov models have been very successful [19][8][2][3][18]. The accuracy of these methods reaches around 80%. An excellent review on the methods for protein secondary structure prediction has been published by Ross [25].

Recently, there has been an increase in pattern based approaches for protein secondary structure prediction for their high accuracy values that are well above 80% accuracy. Among this machine learning methods SVMs, decision trees and random forests have been attracting a lot of attention. In this paper, we propose a new algorithm that adapts a graph theory approach combined with random forests for the secondary structure prediction problem and feature selection. In section 2.2 we give a brief introduction to random forests.

2.3 Random Forests

Random forests are proposed by Leo Breiman [5]. Random forests are a combination of decision trees and each tree is grown from a randomly sampled set of the training data. Each of the classification trees is built using a bootstrap sample of the data. Each tree outputs a class for a given test data and the test data is labeled with the class that has the majority of the votes from these trees. Given M features in a training set, for each decision tree in the random forest the best splitting feature is determined by from a randomly selected subspace of m features at each decision node. The optimal value of m is usually the square root of M; however this m value also depends on the strength and correlation of the trees. The user has to specify the m value accordingly.

Random forests use both bagging and random variable selection for tree building. There is no pruning. Bagging and random variable selection result in low correlation of the individual trees which yield to better classification. Random forests do not overfit and show comparable results to other machine learning approaches such as SVM. It is a robust method concerning the noise and the number of attributes Generated forests in random forests can be saved for future use on other data.

2.4 Feature Selection

Analysis with a large number of variables requires a large amount of memory and computation time. The problem of selecting a subset of relevant features in a large quantity of data is very important. Feature selection is a process commonly used in machine learning, where a subset of the features available from the data is selected for the learning algorithm. Feature selection is often necessary where it is computationally infeasible to use all available features. The benefits of feature selection reduce training and storage requirements. Also, a good feature selection mechanism can improve the classification by eliminating noisy or non-representative features.

There has been a lot of research on feature selection. Birzele and Kramer [4] have used a new representation for protein secondary structure prediction based on frequent patterns which gives competitive results with the current techniques. Shi and P. N. Suganthan [26] investigated feature analysis for the prediction of the secondary structure of protein sequences using support vector machines (SVMs) and K-nearest neighbors algorithm (KNN). They applied feature selection and scaling techniques to obtain a number of distinct feature subsets. Their experimental results show that the feature subset selection improves the performance for both SVM and KNN.

Kurgan and Homaeian [20] describe a new method for prediction of protein secondary structure content based on feature selection and multiple linear regression. The application of feature selection and the novel representation results in 14-15% error rate reduction when compared to results when normal representation is used. Their prediction tests also show that a small set of 5-25 features is sufficient to achieve accurate prediction for helix and strand content for non-homologous proteins Karypis proposes a new encoding scheme and better kernels for protein secondary structure problem [18]. In the proposed new coding scheme both position-specific and non-position specific information is combined for the representation of each protein sequence. In this paper, we compared this new encoding scheme with many different encoding schemes and presented the results.

Su *et al.* [28] have used condensed position specific scoring matrix with respect to physicochemical properties (PSSMP) on the prediction accuracy, where the PSSMP is derived by merging several amino acid columns of a PSSM sharing a certain property into a single column. Their experimental results show that the selected feature set improves the performance of a classifier built with Radial Basis Function Networks (RBFN) in comparison with the feature set constructed with PSSMs or PSSMPs that adopt simply the conventional physicochemical properties. In order to get an effective and compact feature set on this problem, they propose a hybrid feature selection method that inherits the efficiency of uni-variant analysis and the effectiveness of the stepwise feature selection that explores combinations of multiple features. They decompose each conventional physicochemical property of amino acids into two disjoint groups which have a propensity for order and disorder respectively. Then, they show that some of the new properties perform better than their parent properties in predicting protein disorder.

In this paper, we have applied a different approach from that of Su et al, to condense the PSSM matrix. We propose an algorithm that used a graph theory approach for feature selection. First, we apply this algorithm on BLOSUM62 matrix and then based on the feature set produced by the algorithm; we use this feature set for condensing the PSSM matrix. The details of our method are presented in the next section.

3 Methods

3.1 New Algorithm for the Prediction of the Secondary Structure

This study attempted to reduce the feature space of the dataset using a graph theoretical approach. Even though, graph theory concepts have been around for more than a century, its concepts are just newly being explored for applying to biology [6][30]. The clique search algorithm was applied to find all the cliques with the different threshold values. We used Niskanen's and Ostergard's original implementation of the *Cliquer* version 1.1 [21]. The code *Cliquer* is a set of C routines for finding cliques in an arbitrary weighted graph. It uses an exact branch-and-bound algorithm recently developed by Östergård [23].

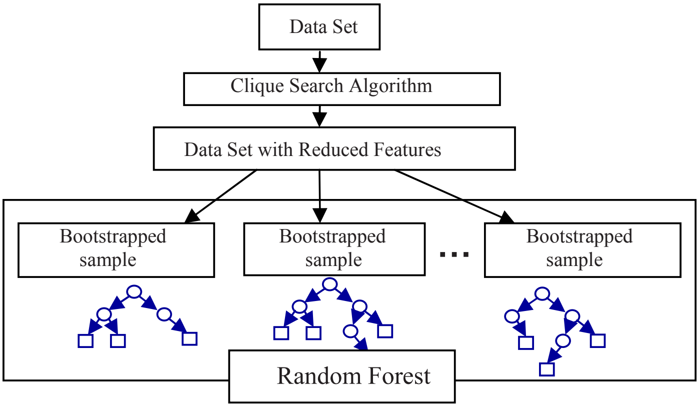


Fig. 1. New model for the prediction of the secondary structure

Next, based on the newly designed algorithm, final cliques were determined. By merging the vertices within the same clique into one, the original feature space is reduced. Finally this reduced feature set was applied to random forests and the performance was compared with the unreduced counterpart. In Fig. 1, the whole picture of this model is presented.

3.2 Encoding Schemes of the Data

The two matrices such as Blosum62 and PSSM were applied alone or combined with a feature reduction scheme. BLOSUM62 matrix is a measure of differences between two distantly related proteins. The values in the BLOSUM62 matrix represent the possibility that a given amino acid pair will interchange with each other in the evolutionary process. The position-specific scoring matrix (PSSM) generated by PSI-BLAST uses position-specific scores for each position in the alignment. Highly conserved positions have high scores and weakly conserved positions have low scores close to zero. Since each of these coding schemes captures different aspects of the properties of the amino acids, the combinations of these two different encodings would be more informative.

The above encoding profiles were generated based on the sliding window scheme. In the sliding window scheme, a window becomes one training pattern for predicting the structure of the residue at the center of the window. The optimal window size of the sliding window scheme was set as 13 based on the previous research [15]. To reduce the noise in the training data and to minimize the memory requirement for training, the feature set was reduced based on the clique search algorithm. This approach is described in detail in the next section.

3.3 Feature Reduction Based on Cliques

A clique in an undirected graph G is a set of vertices V such that for every two vertices in V, there exists an edge connecting the two. The subgraph induced by V is a complete graph. The size of a clique is the number of vertices it contains. The maximum clique problem is to find the largest clique in a graph.

The BLOSUM62 matrix of this study can be represented as a graph which consists of 20 different vertices. The edges among these 20 vertices can be introduced by applying different threshold values to the BLOSUM62 matrix. This study attempted to reduce the feature size by obtaining the cliques which occur commonly in different threshold values and by merging the vertices within the same clique. This process can be divided into the following three steps. The first step is converting the matrix into the adjacency matrix based on different threshold values ranging from -2 to 2. Each cell of the adjacency matrix has a value '1' if there is an edge between two vertices and a value '0' if there is no edge between them based on different threshold values. The second step is applying the clique search algorithm to each of these adjacency matrices. The third step is scanning through all the cliques obtained from each matrix and finding the common cliques. The cliques of size 2, 3 or 4 vertices (n-mer) which share at least one physico-chemical property (polar, hydrophobic, or aromatic etc) were considered for final decision. The common cliques were determined by counting the same vertices (n-mer) in each clique. Based on our voting scheme, three most commonly occurring n-mers were found by our proposed algorithm. These were merged into one-mers such as follows:

- Q E → E
- I L M → L
- H F Y → Y

```
Input: Blosum62 matrix B
        Threshold set T    $T = \{-2, -1, 0, 1, 2\}$
        Physico-chemical property sets P    $P = \{P_1, P_2, \ldots, P_8, P_9\}$
Output: Common_Clique_Set C

Process:
    FOR each threshold i of T
        Adj_Matrix$^i$ = Create_adjacency_matrix (B)
    END FOR

    FOR each adjacency matrix Adj_Matrix$^i$
        Clique_Set$^i$ = Find_all_cliques (Adj_Matrix$^i$ )
    END FOR

    FOR each clique set Clique_Set$^i$
        FOR each clique j    $j \in$ Clique_Set$^i$
            if size_of(j) equals to 2 or 3 or 4
                FOR each $P_i \in P$
                    if $j \subseteq P_i$
                    count++
                END FOR
                Save the count into count_array
        END FOR
    END FOR
Common_Clique_Set C = Vote_and_Find_Top_Three(count_array)
```

Fig. 2. Common clique search algorithm

Table 1. Physico-chemical property set

Set P	Physico-chemical properties	Amino acids in each set
P_1	Small	A, C, D, G, N, P, S, T, V
P_2	Hydrophobic	A, C, F, G, H, I, K, L, M, T, V, W, Y
P_3	Polar	C, D, E, H, K, N, Q, R, S, T, W, Y
P_4	Tiny	A, C, G, S
P_5	Aliphatic	I, L, V
P_6	Aromatic	F, W, Y
P_7	Charged	D, E, H, K, R
P_8	Positive	H, K, R
P_9	Negative	D, E

The pseudocode of this algorithm is given in Fig. 2. The physico-chemical property sets P in the pseudocode is described in Table 1.

The BLOSUM62 matrix can be reduced to the size of 15x15 based on the above compression. By applying the same reduction, the PSSM can also be compressed into Lx15. Here, L is the sequence length of the protein.

3.4 Training and Testing

The commonly used RS126 set was applied to compare the results with previous studies. The RS126 data set was proposed by Rost & Sander [27] and is known to be a non-homologous set which shares less than 25% sequence identity. The random forests algorithm performs bootstrap test with the training data. In other words, one-third of the instances are left out in the construction of the k^{th} tree and these are applied for classification. Therefore, in random forests we do not need to perform a cross-validation. Nor we need to save a separate test set to obtain an unbiased accuracy values. However, the current study applied two thirds of the original data for training and one third for testing to confirm the result obtained from the training data.

3.5 Parameter Optimization

In the random forests program, the only parameter to be optimized is the number of features called m_{try} that are randomly selected at each node [8]. As a rule of thumb, the author suggested that it could be set as the square root of the number of whole features. Including this value, this study tested different m_{try} values to find the optimum value.

3.6 Binary Classifiers

Six binary classifiers, such as three one-versus-rest classifiers (H/~H, E/~E and C/~C), and three one-versus-one classifiers (H/E, E/C and C/H) were created based on the previous study [16]. Here, the name ‘one’ in one-versus-rest classifier refers to positive class and the name ‘rest’ means negative class. Likewise, the name ‘one’s in one-versus-one classifier refers to positive class and negative class respectively. For example, the classifier H/~H classifies the testing sample as helix or not helix and the classifier E/C classifies the testing sample as sheet or coil.

4 Results

4.1 Parameter Optimization

Table 2 presents the result of applying different m_{try} values (the number of features randomly selected) based on the Blosum62 and reduced PSSM concatenated encoding scheme. In the second column of the table, the value 22 is obtained from the square root of the whole dimension of the feature: the whole dimension is (20+15) * 13 = 455. As it can be seen from the table, the accuracy values are almost same even though we choose the larger m_{try} values. This means that the square root value is almost the optimal.

Table 2. Comparison of different m_{try} values

Binary classifier	Accuracy (%) for different m_{try} values			
	22	**50**	**100**	**200**
H/~H	82.2 85.1	82.1 85.6	83.3 85.6	82.1 85.1

4.2 Encoding Scheme Optimization

Table 3 shows the result obtained by applying different encoding schemes into the random forests. Two different accuracy values are displayed. The first row is obtained by bootstrap test on the training data and the second row is on the test data. As it can be observed from the table, both the reduced Blosum62 matrix and the reduced PSSM encodings present equal level of accuracy values when compared with the unreduced counterparts whether applied alone or applied with concatenated form. This result proves that there is no information loss from the feature reduction and our algorithm for this reduction works properly. Among the all different encoding schemes, the reduced PSSM encoding shows the best performance. The reduced PSSM encoding performs comparable to the concatenated encoding of reduced PSSM and BLOSUM62 matrix. Besides reduced PSSM shown in the last column has 13*15=195 features whereas unreduced PSSM 13*20=260 features. This means that ~25% feature reduction is achieved using by our algorithm while still achieving high accuracy.

Table 3. Comparison of different encoding schemes for H/~H binary classifier

	PSSM	**Reduced PSSM**	**BLOSUM**	**Reduced BLOSUM**	**PSSM+ BLOSUM**	**Reduced PSSM+ BLOSUM**	**Reduced PSSM+ Reduced BLOSUM**
H/~H	82.3 85.5	82.5 85.7	76.9 80.7	77.2 80.8	82.3 85.1	82.2 85.1	81.7 85.0

In Table 4, the all six binary classifiers are tested based on BLOSUM and PSSM combined encodings. Once again, it can be observed that the reduced PSSM encoding has almost the same performance as the unreduced counterpart against all six binary classifiers.

Table 4. Accuracy results with BLOSUM+PSSM encoding

Binary classifiers	Accuracy for PSSM+BLOSUM	Accuracy for reduced PSSM+BLOSUM	Accuracy for reduced PSSM
H/~H	82.3	82.2	82.5
	85.1	85.1	85.7
E/~E	83.9	83.7	84.0
	81.1	81.1	81.0
C/~C	76.1	75.7	76.3
	75.5	74.9	75.6
H/E	85.2	85.3	86.5
	83.1	82.7	84.0
E/C	79.5	78.9	80.6
	78.7	78.6	80.3
C/H	82.0	82.1	82.2
	83.2	82.9	83.3

4.3 Time Comparison

Table 5 shows the execution times of reduced PSSM encoding scheme versus PSSM+BLOSUM encoding scheme with different number of trees. Our proposed encoding scheme reduced PSSM has a faster execution time. Also, when using 2000 trees, PSSM+BLOSUM encoding scheme didn't run after some time due to its high dimensionality whereas reduced PSSM encoding could run. These results show that reduced PSSM encoding could be used to reduce space and time complexity drastically where the data dimensionality is very high.

Table 5. Comparison of execution times for reduced PSSM vs. PSSM+BLOSUM

Tree size	Encoding Scheme	
	PSSM+BLOSUM	Reduced PSSM
100	25min 58.9s	5min 53.7s
500	153min 50.6s	31min 8.5s
1000	267min 31.9s	66min 15.8s
2000	–	124min 24.7s

5 Conclusion

In this paper, we proposed a novel algorithm for feature selection based on cliques and evolutionary information of proteins. We tested our algorithm using random forests and different encoding schemes for the secondary structure problem in proteins. These algorithms were tested on a condensed and non-condensed data set. We found out that the prediction accuracies for both data sets were almost similar. These results show that a significant amount of space and time can be saved while still achieving the same and high accuracy results by using a subset of the features when these features are carefully selected. These results show that it is important to select features from data that are more significant for training and testing instead of using all the features set. Also, using our novel algorithm, we achieved about 25% reduction in space and time. In the future, we plan to enhance our algorithm using different approaches from graph theory such as vertex cover and machine learning algorithms such as SVM. We will also test different evolutionary matrices and apply different encoding schemes.

Acknowledgments. This research was supported in part by the U.S. National Institutes of Health under grants R01 GM34766-17S1, and P20 GM065762-01A1, and the U.S. National Science Foundation under grants ECS-0196569, and ECS-0334813. This work was also supported by the Georgia Cancer Coalition (GCC) and the Georgia Research Alliance. Gulsah Altun and Hae-Jin Hu are supported by Georgia State University Molecular Basis of Disease Fellowship.

References

1. Altschul, S. F., Madden TL., Schaffer AA., Zhang J., Zhang, Z. , Miller, W., Lipman DJ.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, Nucleic Acids Research, Vol 25, no.17 (1997) 3389-3402
2. Altun, G., Hu, H.-J., Brinza, D., Harrison, R.W., Zelikovsky, A. and Pan, Y.: Hybrid SVM kernels for protein secondary structure prediction, *Proc. IEEE Intl* Conf on Granular Computing (GRC 2006), 762-765.
3. Aydin Z., Altunbasak Y., and Borodovsky M.: Protein secondary structure prediction for a single-sequence using hidden semi-Markov models, *BMC Bioinformatics*. (2006) Vol. 7, 178
4. Berman, H., Henrick K., Nakamura H. and Markley J.L.: The worldwide Protein Data Bank (wwPDB):ensuring a single, uniform archive of PDB Data
5. Birzele F. and Kramer S.: A new representation for protein secondary structure prediction based on frequent patterns. Bioinformatics (2006) Vol. 22, no.21, 2628-2634
6. Butenko, S., Wilhelm, W.: Clique-detection models in computational biochemistry and genomics. European Journal of Operational Research (2006). To appear. Available online at http://www.sciencedirect.com/
7. Breiman, L.: Random Forests. Machine Learning Vol. 45. no. 15-32
8. Breiman, L. and Cutler, A.: Random Forest, http://www.stat.berkeley.edu/~breiman/RandomForests/cc_software.htm
9. Bystroff, C., Thorsson, V., Baker, D.: HMMSTR: a Hidden Markov Model for Local Sequence Structure Correlations in Proteins. *J Mol Biol. (*2000) Vol. 301 173–190.

10. Chou, P.Y, Fasman, G.D.: Prediction of protein conformation. Biochemistry. (1974) Vol. 13, no.2, 222–245.
11. Efron, B. Tibshirani, R.: An Introduction to the Bootstrap. Chapman and Hall, New York (1993).
12. Fleming, P.J., Gong, H. and Rose, G.D.: Secondary structure determines protein topology. *Protein Science*, Vol. 15, (2006) 1829-1834
13. GOR Garnier, Osguthorpe and Robson.: Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. J. Mol. Biol., (1978) Vol. 120 97-120
14. Henikoff, S. and Henikoff, J. G.: Amino acid substitution matrices from protein blocks. Proc. Natl. Acad. Sci. (1992) Vol. 89, 10915-10919.
15. Hu, H., Pan, Y., Harrison, R. and Tai, P.C.: Improved protein secondary structure prediction using support vector machine with a new encoding scheme and an advanced tertiary classifier," *IEEE Trans. NanoBiosci.*, (2004) Vol. 3, 265
16. Hua, S. and Sun, Z.: A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach," *J. Mol. Biol*, (2001) Vol. 308, 397-407
17. Jones, D.T.: Protein secondary structure prediction based on position-specific scoring matrices. *J Mol Biol. (*1999) Vol. 292:195–202.
18. Karypis, G.YASSPP: better kernels and coding schemes lead to improvements in protein secondary structure prediction.Proteins. (2006) Vol. No. 64(3):575-86
19. Kloczkowski, A., Ting KL, Jernigan RL, Garnier J. Combining the GOR V algorithm with evolutionary information for protein secondary structure prediction from amino acid sequence. *Proteins. (*2002) Vol. 49, 154-166.
20. Kim, H., Park, H.: Protein Secondary Structure based on an improved support vector machines approach. *Protein Eng. (*2003)
21. Kurgan, L. and Homaeian, L.: Prediction of Secondary Protein Structure Content from Primary Sequence Alone-A Feature Selection Based Approach Machine Learning and Data Mining in Pattern Recognition Vol. 3587 (2005) 334-345
22. Niskanen, S. and Östergård, P.R.J.: Cliquer User's Guide, Version 1.0, Communications Laboratory, Helsinki University of Technology, Espoo, Finland, Tech. Rep. T48, (2003).
23. Östergård, P.R.J., A fast algorithm for the maximum clique problem, Discrete Applied Mathematics, (2002) Vol.120,0 no.1-3, 197-207
24. Przytycka, T., Aurora, R., Rose, G.D.: A protein taxonomy based on secondary structure. *Nature Structural Biol*. Vol. 6 (1999) 672-682.
25. Przybylski, D, Rost, B.: Alignments grow, secondary structure prediction improves. *Proteins.* Vol. 46 *(*2002) 197-205.
26. Rost, B.: Rising accuracy of protein secondary structure prediction. In: Chasman D.,editor., *Protein structure determination, analysis, and modeling for drug discovery.* New York: Dekker, (2003) 207-249
27. Rost, B. and Sander, C.: Prediction of protein secondary structure at better than 70% accuracy, *J. Mol. Biol.*, (1993) Vol. 232, 584-599.
28. Shi, S. Y. M. and Suganthan, P.N.: Feature Analysis and Classification of Protein Secondary Structure Data, In Lecture Notes in Computer Science, Vol.2714, (2003) 1151-1158, Springer-Verlag Berlin, Germany
29. Su, C.-T., Chen, C.-Y. and Yu-Yen, Ou.: Protein disorder prediction by condensed PSSM considering propensity for order or disorder, *BMC Bioinformatics*, (2006) Vol. 7., 319
30. Vishveshwara, S., Brinda, K.V., Kannan, N.: Protein Structure: Insights from Graph Theory Jl Th Comp Chem. (2002) Vol. 1., 187-211

DNA Sites Buried in Nucleosome Become Accessible at Room Temperature: A Discrete-Event-Simulation Based Modeling Approach

Amin R. Mazloom[1], Kalyan Basu[1], Subhrangsu S. Mandal[2], Mehran Sorourian[3], and Sajal Das[1]

[1] CReWMaN Lab, Department of Computer Science and Engineering
[2] Department of Chemistry and Biochemistry
[3] Department of Biology
The University Of Texas at Arlington, Arlington TX 76019-23015, USA

Abstract. Conformation of a canonical nucleosome inhibits the direct access of the binding proteins to portions of nucleosomal DNA. Nucleosome dynamics establish certain pathways through which nucleosome gets remodeled (spontaneously, covalently or non-covalently) and the buried DNA sites become accessible. Currently for most pathways no single model is available to capture the temporal behavior of these pathways. Plus traditional diffusion-based models in most cases are not precise. In this work we have given a systematic overview of such pathways. Then, we manipulate the probability of a binding site on array of N nucleosomes and chromatin of length G base pairs . We further identify three of the widely accepted thermal-driven (passive) pathways and model those based on stochastic process and the Discrete-Event-Simulation. For the output of the models we have sought either the site access rate or the sliding rate of the nucleosome. We also show that results from these models match the experimental data where available.

1 Introduction

DNA access is the key to cell protein machinery both in prokaryotes and eukaryotes. The long DNA chain of the eukaryotes use a systematic hierarchical compression. In the lowest compaction level the genetic material comprises arrays of coiled DNA around globular octamer of cationic nucleus proteins (histone) [1]. Each of these array elements is referred to as nucleosome and the chain of the ∼1.65 left-handed superhelical turn is known as chromatin [2][3].

A nucleosome component consists of ∼200 base pairs (bps) (∼147 nucleosomal-DNA bps), ∼ 153 linker-DNA bps) in euchromatin which is the most permissible chromatin conformation in gene expression [1]. Higher order DNA structures are also available in the form of 30nm chromatin fiber [4] and heterochromatin [15], all of which encapsulate the multi-billion base pairs of higher organism in the nano-metric volume of nucleus. Massive research in the past thirty years on the chromatin structure and dynamics has revolutionized our knowledge on chromatin and its dynamics, yet much of the actual mechanistic is still far from completely understood. The experimental data of in-vivo and in-vitro assays have played an important role in the success of biologists

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 601–614, 2007.

and biochemists in one or more of following aspects: (a) explaining the nature of the nucleosome related bio-processes, (b) proposing rational speculations where no explicit indication of actual mechanism could be observed, or (c) providing some useful empirical data in different domains (time, concentration , gene expression level , etc.) that would help further discoveries and validations. Scientists are now convinced that, in order to have a better understanding of a biological process they have to understand that at the cell level. The fundamental challenge in such a system-level understanding is the complexity of the molecule bodies and their interactions at the cell level, and this complexity increases manifold in higher scales. An example of that is the interaction of cell ensembles in a tissue. For this purpose one needs to have individual models for lowest level processes in place and very finely define their interaction in hierarchical order. We have previously defined the concept of stochastic discrete event simulation of biological systems iSimBioSys [5] and introduced this lower biological function modeling method. We further have shown how these lower level models can be used to build a complex biological system. The models that we have introduced includes: (1)DNA-protein binding [27], (2) Protein-ligand docking [6], (3) Cytoplasmic reaction [26], and (4) Molecule intake by bacteria [28]. In this paper we use the same concept to develop parametric understanding of nucleosome dynamics specifically the passive pathways.

The dynamics of nucleosome establishes certain pathways for accessing the hindered portions of nucleosomal DNA (nDNA). From the energetic point of view, a pathway that requires energy for its progression is called an *'Active pathway'* and if it is spontaneous, is referred to as a *Passive pathway*. For the latter which is the focus of this paper we incorporate spontaneous *unwrapping and rewrapping* and also *nucleosome sliding* pathways. We implement two mechanisms for nucleosome sliding: twist-defect [7] and planar-bulge inchworm [8]. Also we try to benefit our modeling approach from the virtue of any of biophysical and/or biochemical models that are available elsewhere and fit appropriately with our schema. In this work, we first find the probabilities of finding a motif of length n elsewhere genome wide. Second, we propose a stochastic model for *unwrapping and rewrapping* and come up with a closed form solution. Thirdly, for twist-defect and planar-bulge inchworm we propose two models both of which are fully stochastic, where the first model is sequence specific. The rest of the paper is organized as follows: We briefly discuss the important aspects of nucleosome dynamics in Section 2. Motif access pathways and the pathway model is presented in Section 3. Section 4 discusses the stochastic component of the model and proposes the abstracts for the species. In Section 5, two model each corresponding to one mechanisms of DNA access is proposed. Numerical analysis and validation is given in Section 6. Conclusions are drawn in the last section.

2 Nucleosome Dynamics

Nucleosome is not a monolithic static assembly. In literature the dynamics of nucleosome is segregated into three categories; however there is a belief that conformational fluctuation is the forth category. Therefore, a nucleosome exhibits at least four dynamics: (1) compositional alternation, (2) covalent modification, (3) translational repositioning, and (4) conformational fluctuation. Compositional alternation is done by some remodeling enzymes to promote gene activation [12]. For example, although canonical nucleosomes are deposited during cell replication, H2A variant H2A.Z are highly enriched in promoter area which is deposited by SWR1 chromatin-remodeling complex [11]. Post translational modifications including acetylation, methylation, phosphorylation and ubiquitination [13] are among the covalent modifications that can destabilize the histone cores and exploit DNA access to the biological processes. ATP-dependant remodelers use energy derived form ATP hydrolysis to loosen the contacts between the coiled DNA and the histone core. All these remodelers have at least one ATPase domain that provides the energy necessary to alter the nucleosome conformation. For instance, it takes less than one second for SWI/SNF to remodel a nucleosome[1]. The nucleosome remodeling is a catchall for different mechanisms proprietary to each family of remodellers or an individual protein complex. Some of the biochemical activities that the remodeling mechanisms evolve from include: creation of translational positions of histone octamer, ejection and spacing of nucleosomes, histone octamer transfer, creation of remodeled di-nucleosome species and altered restriction enzyme access [10]. In translational repositioning the bp position of core particle in the genome changes to enhance the target site access. This process could happen both intrinsically or by the aid of remodellers. Conformational fluctuation is a periodic minor change to the conformation of a canonical nucleosome happens in the room temperature and will be further elucidated in successive sections.

3 Motif Access Pathways

Coiled structure of DNA in nucleosome confined the accessibility of the portion of DNA where the minor groove of helical turn faces the histone core. In euchromatin, passive and active pathways make a buried target motif (site) accessible by employing one or more of the nucleosome Conformation Alternator Families (CAF). Main CAFs include: (1) Thermal spontaneous conformational fluctuations [3] [8], (2) ATP-dependent chromatin remodeling [9], and (3) covalent nucleosome modifications [10]. Each CAF might have different classes and mechanisms, where they could work individually or in a concerted fashion. In the latter, they facilitate each others' functions, allowing multiple options to be considered during the evolution of a specific pathway for an individual motif access. The eventual goal is to have a properly structured template with all compulsive pre-initiation complexes, available in a timely manner.

3.1 Pathway Model

In eukaryotes assembly of transcription apparatus and associated basal transcription comprises access of promoter by several other TF target sites. Every access pathway is distinguished based on the mechanism that the pathway employs. Each of these pathways could further be broken into complex specific pathways. Figure 1 depicts a set of plausible mechanism level motif access pathways part of which is the focus of our investigation. In this paper we just concentrate on passive pathways that is shown in light

[1] The mechanisms implemented by remodelers and their pathways are beyond the scope of this paper. Currently in CReWMaN, we are conducting another work to model the pathways of SWI/SNF and ISWI remodelers.

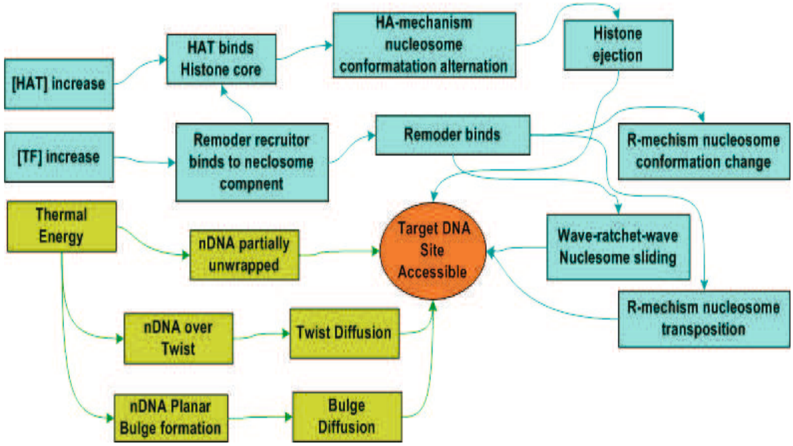

[HAT] increase
HAT binds Histone core
HA-mechanism nucleosome conformatation alternation
Histone ejection
[TF] increase
Remoder recruitor binds to neclosome compnent
Remoder binds
R-mechism nucleosome conformation change
Thermal Energy
nDNA partially unwrapped
Target DNA Site Accessible
Wave-ratchet-wave Nuclesome sliding
R-mechism nucleosome transposition
nDNA over Twist
Twist Diffusion
nDNA Planar Bulge formation
Bulge Diffusion

Fig. 1. nDNA accessibility mechanism pathway; each pathway starts with one trigger event; *HA-mechanism nucleosome alternation* block represents the mechanism implemented by Histone acetylation; *R-mechanism nucleosome conformation alternation* and *R-mechanism nucleosome transposition* depicts two classes of remodeling mechanism that derive the remodeling process

green in Figure 1. Each block in a pathway is an event and since the cell is a random environment due to the stochastic resonance [16] we will treat each pathway as a series of uncorrelated random events in the time and space and offer a probability distribution for them. The collaborative effect of these stochastic events in a concerted fashion will lead to motif accessibility. In distinguishing the series of events in a pathway we will assume the discrete event simulation methodology. Thus, parallel events are allowed to be executed as long as they do not have resource access collision among them, otherwise they need to be serialized [17].

4 Assumptions and Stochastic Components

In order to manipulate the temporal behavior of each under study pathways of Figure 1, we need to have execution time of individual components (event) and the respective probability distribution of each event. For that purpose we need to make certain rational assumption without losing the generality and introduce the abstract bodies that can represent physical topology of the individual participant of the process. The abstract bodies that we shall use along the paper are as follows:

DNA: The DNA model that we are using is the rod-like model from [19]. Such a DNA model is formed from beads that are inter-connected by harmonic spring in a spatially harmonic fashion. This model defines three dynamics for the DNA: longitudinal, rotational and bending. These dynamic will further be used in the mechanical models.

Nucleosome: Nucleosome consists of two components, the histone core and nDNA. Throughout this paper, histone core is assumed to be a rigid cylinder with no structural alternation. The nucleosomal DNA is attached to the histone core at 7 absorption points per super-helical turn, starting from the dyad [23] axis down to end of core cylinder. These contact points are perpendicular to the super helix axis of nucleosome. In [20] the nucleosomal DNA is mapped into Frenkel-Kontorova chain and the total energy of the canonical nucleosome in equilibrium is manipulated based on three energy components: elasticity energy (E_l), docking point energy (E_d), and sequence energy (E_s). We shall follow the same energy concept for manipulating the energy barriers in the model where applicable.

Proteins: In our processes, proteins appear in two forms: (a) single protein which is one polypeptide of amino acids molecule within its proper tertiary structure (e.g. a single transcription factor recruiter), or (b) protein complex that is made of number of protein molecules in their quaternary structure prior to binding to the target site, such as RNAP II holoenzyme or SWI/SNF remodeling complex. In either case we abstract the molecules with a spherical body. For the former we approximate the diameter of the sphere by the average width of the protein, where in latter this is done through averaging over width of the complex.

Chromatin Wide Collision

For any binding to happen a priori is to have collision between the parties, for our work we follow the collision theory concept of an infinitesimally small time $\varDelta t$. We assume that the continuous nucleosome chain inside the nucleus is not repositioning. Also since the nucleosome array are available in the form of continuous chain, therefore their population density is non-uniformly distributed and is directly proportional to the density of the chain . For this purpose, we divide the nucleus into s equal volume partitions $\nu_s = \nu/s$ where each partition will have $\rho_i = N_i/\nu_i$ nucleosome density, N_i being the number of nucleosomes in partition i and ν_i as the respective partition volume. We further assume that the second colliding party, e.g. TF or RNP II, is uniformly distributed inside the nucleus. Hence, one can find probability of collision between a TF and any nucleosome component in partition i druring $\varDelta t$ from the same way as calculated in [26] from:

$$p^i_{col} = \pi(r_p + 1/2(r_{nuc} + r_\ell))^2 \sqrt{\pi 8 k_b T} \varDelta t [TF] m_p^{-1/2} \nu_s \tag{1}$$

In Eqn.(1), r_{nuc} and r_ℓ are the nucleosome and linker DNA (lDNA) radiuses, r_p and m_p are TF radius and mass, k_b is the Botzmann constant, T is kelvin temperature, and [TF] is the transcription factor concentration.

In the partition borders, the chain structure of chromatin enforces a close correlation among the nucleosome population distribution in the local neighboring. Hence the effect of choosing the distribution form either in the final results is negligible. Considering Eqn.(1), in order to get the collision probability anywhere in the nucleus we can use, $p^\nu_{col} = 1/s \sum_i \rho_i p^i_{col}$. Also, the probability just to collide with a specific nucleosome is:

$$p^N_{col} = \frac{p^\nu_{col} v_{nuc}}{\nu} = \frac{p^\nu_{col} r^2_{nuc} L_{nuc} \lceil G/200 \rceil}{1/2(r_{nuc} + r_\ell)^2 (l_{nuc} + l_\ell)} \tag{2}$$

Here, l_{nuc} is the nucleosome length where l_ℓ is the lDNA length and G is the bp length of genome.

Each complete helical turn (360°) consists of $\sim$10.2 bps [8] in the nucleosome relax condition, where for our calculation we make rough assumption of 10 bps per helical turn. The outer face of each symmetric half of the helical turn forms 6 DNA histone core association points, which along with an additional contacts between the N terminal of $\alpha N - helix$ and the N termini of histone H3 tail is referred as Super-Helical- Locations (SHL) [8],(see Figure 5). Therefore, in each nucleosome we have 14 association points. Also we know that DNA packaging and occlusion causes on each helical turn, inhibits the access to those bps that face the histone octamer where the minor groove of DNA contacts the core particle, at SHLs, this structural conformation is further elucidated in [23]. Having these in mind, we derive few useful probabilities. Probability that a motif i ($i \leqslant 10$) bps lays in one helical turn, p_h, is:

$$p_h = \frac{14(10 - i + 1) \times L}{(L - 147 + 1)(147 - \ell)} \tag{3}$$

In the above expression, L is the length of DNA in bps and ℓ is the average linker DNA length. In each helical turn coiled on canonical nucleosome only half of the base pairs are accessible; therefore the probability of a motif of length i, ($i \leqslant 5$) to be directly accessible is: $p_m^i = p_h \frac{5-i+1}{10-i+1}$.

Seemingly the probability of having the motif of length i, ($i \leq 147$) in one nucleosome, p_n^i will be: $p_n^i = \frac{L(147-i+1}{(147-\ell)(L-i+1)}$ and if $L \gg i$ then, $p_n^i \approx \frac{147-i+1}{147-\ell}$.

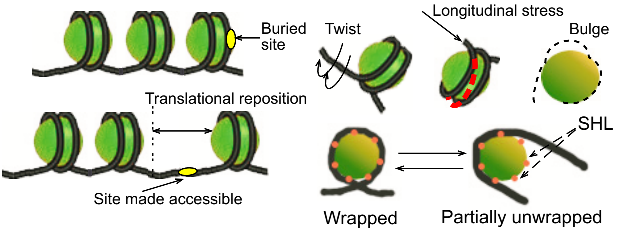


Fig. 2. Left side: The upper image shows the site buried in the nucleosome, and the lower one shows site access made possible through translational repositioning; Right side: The upper shows the dynamics that can lead to translational repositioning of nucleosome, and the lower image plots the partial uncoiling of nDNA that makes a site accessible

Now we make another rational assumption for the structure of the DNA turns around the histone octamer and keep up with that for the rest of the paper: DNA helical turn on entry and exit points of nucleosome have 180° phase difference. On the other hand, if we assume that on entry point DNA enters with major groove and leaves the nucleosome with minor groove, therefor the leading flanking tail of the nDNA is always accessible and the lagging one is doomed. Also we assume all of the motifs that we are working with in this paper shall remain in one super-helical turn. Hence we avoid trapping nucleosome in a DNA knot [18]. This assumption enforces a maximum motif length $i_{max} = 147/2 \approx 73$. Another useful probability will be the probability of accessing a motif of length i on the linker. If $\ell \approx 53$ the average length per linker DNA and $g = 3.5$ the accessible length of lagging flanking nDNA, then the maximum length of motif on the linker DNA, that could be accessed without moving the nucleosome is: $i_\ell = \ell + 5 + g \approx 62$.

Furthermore probability of finding a motif of length i , $i \preceq 63$ in the linker DNA could be written as:$p_{\ell m} = \frac{(\ell_j+9-i+1)\times(N-1)}{L-i+1} = \frac{\ell_j+10-i}{147-\ell}$, where ℓ_j is the lDNA between j^{th} and $j+1^{th}$ nucleosome.

5 Spontaneous Mechanisms of DNA Accessibility

Most of the DNA involved reactions in eukaryotes requires DNA packaging alternation for specie accessing the target site. Amongst mechanisms spontaneous nucleosome alternation is a slow phenomena driven by the thermal molecular energy that happens in the room temperature. In this work, we try to use the discrete-event-simulation approach to model three widely accepted mechanisms of this kind including: partial unwrapping/rewrapping of nucleosomal DNA, twist-defect nucleosome shifting, and planar-bulge inchworm sliding of nucleosome[2].

5.1 Partial Unwrapping/Rewrapping of Nucleosomal DNA

The propensity of nucleosomal DNA (nDNA) to partially unwrap on either end, gives temporal access to the DNA sequence whose readability was obstacled in the canonical conformation. This scaffold is driven by thermal force in a periodic manner [3] [9].

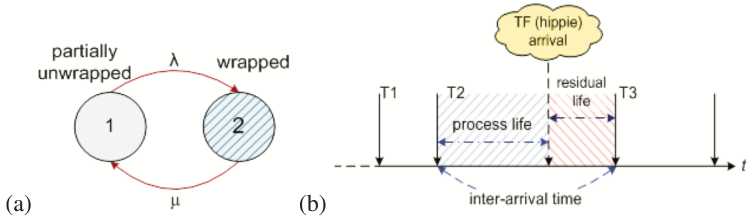


Fig. 3. (a) Projecting the partial unwrapping to a *renewal process*; Upon arrival of TF during state 1 life time there is a chance of accessing the hindered sites; (b) *Process life/residual life concept*: process life lays between the two state arrival time, where the residual life is the interval between present time and the arrival moment of next state

As reported in [3], nDNA remains fully wrapped for $\sim$250 ms then spontaneously becomes partially unwrap, it remains in that state for $\sim 10-50$ ms and then rewraps again. In the latter state, if any of the protein machinery TFs find their target sites on the DNA , which would have been hindered otherwise, binding is highly probable. This scaffold manifests a nature of renewal processes [25], where the active service time of

[2] Proofs of the selected expressions that appear in the subsequent sections are available as supplementary materials online at BioNet web site: *http://crewman.uta.edu/dynamic/bone/publications.htm*

the process is the period in which process reside in state 2 of Figuer 3. We assume this service time is negative exponentially distributed, $p(\tau_1) = e^{-\mu\tau_1}$. Therefore, based on the data in [3] one can say: $\mu \in [20, 100]$ and $\lambda \approx 4$. Considering a general distribution for the arrival rate of the binding specie, in order to receive service (binding) it must arrive when the process is in state 2 and $\tau_2 > 0$, where τ_2 is the residual life of state 2. Using the *m/g/1 queue service model and hippie arrival concept* as in [25], we find $p(\tau_2) = \frac{\mu + e^{-\mu\tau_2} - 1}{\mu^2}$; however we need to have $\tau_2 > 0$, thus:

$$p(\tau_2 > 0) = 1 - p(\tau_2 = 0) = \mu^{-2}(\mu^2 - \mu + 1) \tag{4}$$

From Eqn.(4) and $p_2 = \mu/(\lambda+\mu)$, the probability of state 2, we can find the probability of specie reaches unwrapped nDNA, p_{uw}, in time τ from:

$$p_{uw} = \frac{\mu^2 - \mu + 1}{\mu(\lambda + \mu)} \tag{5}$$

As we see the p_{uw} has an stationary probability and it is justifiable because the two integrations, one in calculating the laplace transform of residual life, and other the implicit integration in Eqn.(4) have averaged over the time.

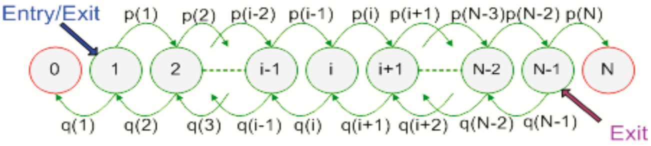


Fig. 4. Markovian Random-walk chain with absorbing states: The absorbing states guaranties two conditions: (a) there will be no re-enter for each walker, and, (b) access to the two exits points are mutually exclusive to every walker

5.2 Twist-Defect Nucleosome Sliding

Introduction of an anomaly into the super-helix conformation of nucleosomal DNA is an alternative mechanism to translocate the nucleosome. The concerted translational and rotational motion of DNA that has high torsional flexibility, leads to injection of this anomaly to the nDNA. Since this phenomena is impelled by thermal energy at the room temperature, it is highly confined only to very efficient torsions. Considering $10pb/turn$ would have $\sim 36°/pb$ in a relaxed nDNA. Therefore twisting or unwinding the DNA double helix by 36° would add or remove one bp to or from the 360° helical turn docked to the histone core between two SHLs [8]. This over-twist/unwound torsion being introduced at an entry or exit point of a nucleosome would have to successfully travel all 147 bps around nucleosome to shift the nucleosome for 1 pb. From this point on we work only with over-twist because the calculation for both are roughly similar. The fluctuation of the twist between each pair of docking points forms a monodimensional random walk (RW). To model this random walk, we define 13 states that resemble the location of the twist between each pair of 14 docking points at arbitrary time t. Our approach is different from the one used in [20] in following aspects: (i) We have used an- isotropic RW where they have isotropic RW (ii) Our Markovian step process has two absorbing state at both ends but they did not include any absorbtion (iii) our model is fully stochastic where as they use deterministic computation in the middle of formulation (iv) we end up with a stochastic process rather than diffusion constant.

In order to move in either directions, twist will require to overcome docking energy (E_d), elastic (E_l) energy, and sequence energy (E_s) barriers. The first two energy barriers are already embedded in the defect potential energy cost expression that is in [20] which is based on the Peierls-Nabarro potential energy concept [21]. By assembling all parts of twist PN equation and integrating the expression given in [21] the average energy of a twist is written as: $\Delta U_{twist} = \frac{1}{12}C + 0.45U_0$. Where in this expression $C \approx (84-120)k_bT$ is the effective *Frenkel-Kontorova* (FK) combined twist and stretch constant, and $U_0 \approx 6k_bT$ is the SHL docking energy. In ΔU_{twist} the an-isotropic bendability of DNA is not included. This propensity of DNA is sequence dependant and the respective energy cost can be calculated from the following equation which is obtained from [20] after proper substitutions:

$$E_s(k) = \sum_{i=1}^{10} -\alpha^s_{i+10(k-1)} cos(i\pi/5) + \beta^s_{i+10(k-1)} \tag{6}$$

Eqn.(6) defines a 10 bp periodic energy field around the histone and linearly assigns two bending energy charges to each dinucleotides, isotropic (β^s_i) and an-isotropic (α^s_i). Here $1 \leqslant k \leqslant 13$ represents the state of defect in Figure 4, (0 and $N = 14$ are the absorbing states). The two approximate bending energy charges per dinuleotide variants, in k_bT unit, are as follows: $(A/T) : \alpha = -43 \quad \beta = -1.5$, $(G/C) : \alpha = 48 \quad \beta = 10$, and $(others) : \alpha = 0 \quad \beta = 3.5$. The defect follows a one dimensional Random Walk (RW) on the nucleosome to exit from either end. If the defect that is inserted at one end (entry point) leaves the nucleosome from the other end (exit point) , this would result a one bp sliding of the nucleosome. We use *Mean First Passage time (MFPT)* to get the defect leaving time from entry point (t_{en}) or exit point (t_{ex}). Using the splitting probability concept given in [22] we can find probability of reaching one site before being absorbed by the other. Therefore, we will have the probability to exit from entry point (p_{en}), and probability to exit from the exit point (p_{ex}) as follows:

$$p_{ex} = \left(1 + \sum_{i=1}^{13} u(i)\right)^{-1} \quad \text{where} \quad u_j(i) = \begin{Bmatrix} \mu_i u(i-1), & \text{for } i > j \\ \mu_i, & \text{i=j} \end{Bmatrix} \tag{7}$$

$$p_{en} = \frac{1 + \sum_{i=2}^{13} v(i)}{1 + \sum_{i=1}^{13} v(i)} \quad \text{where} \quad v_j(i) = \begin{Bmatrix} \mu_i^{-1} v(i+1), & \text{for } i < j \\ \mu_i, & \text{i=j} \end{Bmatrix} \tag{8}$$

In Eqns.(7,8) $\mu_i = p_i/q_i$, where p_i and q_i are probabilities of the defect to move one step to the right or to the left of state i in the unit of time, respectively. Considering the energy barrier, E_s, we can define p_i and q_i as:

$$p_i = \frac{e^{-E_s(i+1)\omega_i}}{e^{-E_s(i-1)\omega_i} + e^{-E_s(i+1)\omega_i}} \qquad q_i = \frac{e^{-E_s(i-1)\omega_i}}{e^{-E_s(i-1)\omega_i} + e^{-E_s(i+1)\omega_i}}$$

$\omega_i = (|E_s(i+1)| + |E_s(i-1)|)^{-1}$ is the projection coefficient that project the energy barrier to $[-1, 1]$, thus, minimize the effect of $exp(\cdot)$ function on the fate of the probability.

Now we apply the concept of Mean First Passage Time (MFPT) for single dimensional RW on a random lattice form [29]. By applying such a method, the pair of exit times, t_{en} and t_{ex} that are the MFPT from exit point and entry point respectively could be manipulated as follows:

$$t_{ex} = \Sigma_{k=1}^{N-1} p_k^{-1} + \Sigma_{k=1}^{N-2} \Sigma_{i=k+1}^{N-1} \Pi_{j=1}^{i} q_i p_j^{-1} \tau_s \tag{9}$$

$$t_{en} = \Sigma_{k=1}^{N-1} p_k^{-1} \Pi_{i=1}^{k} \mu_i \tau_s \tag{10}$$

Here τ_s is one RW step. Also since

$$p_i^s = \prod_{k=0}^{i-1} \frac{q_k}{p_{k+1}} p_1^s \tag{11}$$

is the stationary probability of non-absorbing state i, therefore:

$$p_0^s = q_1 \left(\sum_{i=1}^{N-1} \left(\prod_{k=1}^{i-1} \frac{q_k}{p_{k+1}} \right) + q_1 + \prod_{k=1}^{N-2} \frac{q_k}{p_{k+1}} p_{N-1} \right)^{-1} \tag{12}$$

Arrival rate of the defect, λ_d is confined to: (a) our original assumption that at most one defect could exit in the nucleosome at any moment which is satisfied by p_0 and (b) the energetic probability of a twist formation which is given by Boltzmann factor, $p(\Delta U) = e^{-\Delta U / k_b T}$. Another implicit indication of (a) is $\lambda_d < \mu_d$ where $\mu_d = (p_{ex} t_{ex} + p_{en} t_{en})^{-1}$ is the export rate of defect from either end of the nucleosome.

Having μ_d, Eqns(7,8) and considering (a) and (b), we can give the following upper bound for nucleosome sliding rate to the right ($5^{'}$ to $3^{'}$):

$$\lambda_d \lesssim (p_0 + p_N) \cdot p_{ex} \cdot \mu_d e^{-\Delta U_{twist} / k_b T} \tag{13}$$

We have elucidated the forward sliding ($5^{'}$ to $3^{'}$ wrt dna) of the nucleosome here, likewise approach could also be applied to the reverse sliding ($3^{'}$ to $5^{'}$ wrt DNA).

Our model is more accurate than the one in [20], since they have used isotropic RW and ignored the DNA an-isotropic bendability where we have included the sequence dependant energy directly. Also their diffusion constant was significantly off from real value and they have to use apply the E_s through modified Bessel function to negative exponentially reduce their diffusion constant. However that would not compensable for the contribution of absorbtion states that they have not included on their model which our model comprises. Their intermediate domain switch (to deterministic approach) would induce approximation overhead and consequently less accuracy.

5.3 Planner-Bulge Inchworm Nucleosome Sliding

Formation of bulge is a prominent feature of the nDNA that is cognate with high DNA bendability. If during the final interphase of state 1 (partial unwrap) in section 5.1 a more distal sequence of DNA be pulled in to the nucleosome and be adsorbed to the entry SHL then a bugle is formed. There are two types of bugle: planar and topological bulge. In the planar bulge the entire bulge falls in a same plane, while the topological bulge has a more complex structure where the DNA crosses over itself and creates a twist [23]. In this paper we just consider the small planar bugle since in the passive paradigm it is highly unlikely to have topological bulge. Small bulges are 10 or 20 bps in length, and the reason for having a multiply of 10 bps in their length is to encompass an even number of 360° helical turns. Because otherwise it would impose a phase shift to the nucleosomal DNA which energetically is very costly [8].

Sliding of histone octamer is a consent of bulge formation and annihilation along the nDNA. This resembles the motion of an inchworm creeper around the nucleosome. In [24] they defined an energy barrier for small bulge, then used a one dimensional diffusion to move the bulge around the nucleosome and came up with a diffusion constant. Knowing the motion of the bulge around the DNA is not a deterministic unidirectional motion; therefore, casting a stochastic process to the deterministic domain might further impose inaccuracy and pushes the results away from the real value. In our approach we use the same energy barrier as in [24]. To model the fluctuation of the bulge we use a RW but with some distinction that as the one we applied earlier for twist defect. One major difference in this RW is the isotropy of the steps; therefore in the step process diagram of Fig. 4 for all steps probabilities $p_i = q_i = 1/2$ and E_s would not have any effect in fluctuation of the bulge. Adopting the energy barrier model from [24] after applying proper substitutions, the energy cost $\mathit{\Delta} U$ for a bulge of length l will be:

$$\mathit{\Delta} U_{bulge} = \left(20\pi^4 R_N^4 E_a^5 \sigma k_b T\right)^{1/6} \left(\frac{3.4 l \mathring{A}}{R_N}\right)^{1/3} \tag{14}$$

In (14) R_N is the nucleosome radius, E_a is the docking energy per unit of length, k_b is the Boltzmann constant, T is the absolute temperature, and σ is persistence length of DNA. Persistence length of DNA is the limit beyond which DNA will loose its physical behavior as a pure elastic rod. DNA bend persistence length is ∼150 bps (51 nm).

By applying the same RW algorithms, while $p_i = q_i = 1/2, \quad \forall i \epsilon [1, 13]$, we can find the t_{ex} and t_{en} from the same approach as for twist defect. Again the export rate of the bulge is $\mu_b = (p_{ex} t_{ex} + p_{en} t_{en})^{-1}$. For calculation of arrival rate of bulge λ_b is very similar to the one for twist: $\lambda_b \lesssim (p_0 + p_N) \cdot p_{ex} \cdot \mu_b e^{-\mathit{\Delta} U_{bulge}/k_b T} \times l$. The factor l on the left side of expression is the bulge size. The argument that we made earlier for direction of nucleosome sliding as a result of twist-defect RW stays valid for the bulge as well.

6 Numerical Analysis

In this section we supply some of numerical results that have been derived from the theoretical models we presented earlier. For twist-defect nucleosome sliding, we used two sequences: *PSEN1 bp* ϵ $[-700, -200]$ (human gene responsible for producing precenilin-1 protein), and *AHI1 bp* ϵ $[+181, +681]$ (gene responsible for Jouberin protein in human). Figure 5.(a) shows the variation of nucleosome sliding rate for different

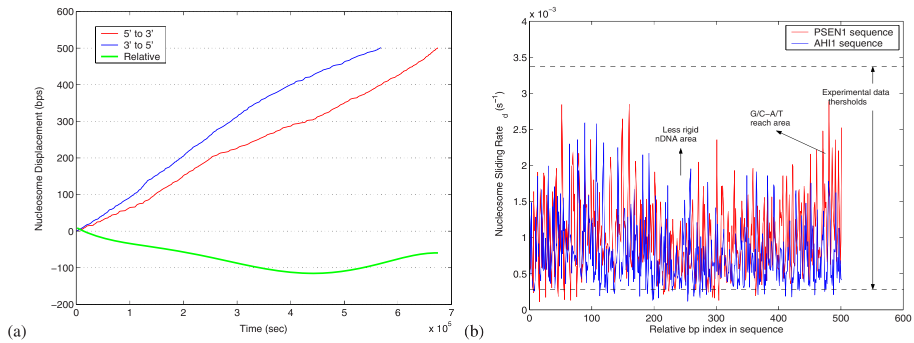


Fig. 5. The twist-defect mechanism used for manipulations in both graph (a) The red line depicts the forward displacement wrt to bp at position -700 in *PSEN1* gene vs. time; The blue line shows the amount of reverse displacement wrt reference bp index vs. time; Green line shows the net displacement of nucleosome, where a negative value indicates the intrinsic sliding of nucleosome tends to slide in reverse direction in this sequence (b) shows the *forward* sliding rates of the nucleosome for *PSEN1_HUMAN* and *AHl1_HUMAN* sequences, the reported experimental thresholds are shown in dashed-lines

base pair in the sequences. As we observe the PSEN1 sequence has larger variation domain. This biologically implies that the sequence has more an-isotropic segments (more populated with A/T and G/C di-nucleotides), hence predicating a lower sliding rate for it. In both sequences the high picks could imply a rotational trap in the DNA where ATP-dependent remodeling might be required and intrinsic sliding gets stock. Due to the granularity of the measurement, finding the sliding rates is a challenging task. Although we could not find any explicit sliding rate for twist defect, we could infer some thresholds form assays in [14]; These thresholds are depicted by dashed lines. The sole theoretical model that we could find to the day of this paper, is a diffusion based deterministic model reported two diffusion constant: $D = 580bp/s^2 \rightarrow \lambda_d \approx 0.79bp/s$ and $D = 10^{-6}bp^2/s \rightarrow \lambda_d \approx 1.2 \times 10^{-8}bp/s$ for isotropic and an-isotropic sequences, respectively. Where the diffusion constant is greater than experimental range by many folds, and for the second one although its closer the lower threshold but still not within the range. Figure 5.(b) depicts the time required for nucleosome to slide on same section of *PSEN1* in forward ($5^{'}$ to $3^{'}$) or reverse ($3^{'}$ to $5^{'}$) direction; However the more interesting result shown in this graph is the relative displacement of nucleosome with respect to the *reference bp* (-700) position which incorporate both forward and reverse repositions. For the bulge inchworm model we got $\lambda_b = 2.14 \times 10^{-7}$ for $l = 10$ *bps* and $\lambda_b = 4.56 \times 10^{-9}$ for a bulge length of 20 bps. The bulge results indicate that within an hour period almost no sliding happens. We concluded that nucleosome sliding through bulge mechanism is a very rare event. We were not able to find any experimental data for the bulge model. We used time step , $\tau_s = 10^{-6}$ throughout our calculations. More numerical results are available for Sections 4 and 5 that are not presented here due to space limitation.

7 Conclusion

We have created a systematic view of the pathways which comprises the synergy of events that effect nucleosome dynamics and procures nDNA accessibility. We have elucidated three of such pathways all of which are passive and tried to stochastically model them. We used the collision theory to manipulate the probability of finding the target nucleosome component in the genome. Also, we have derived the probability of direct accessibility of a target site in the conformation of a canonical nucleosome. For each specie participating in the pathways an abstract model is either proposed or employed from literature. We used the renewal process along with hippie arrival concept to model the spontaneous unwrapping pathway. For twist-defect we used an-istorpic random walk[3] based on the sequence bendability energy barrier to find the MFPT and ultimately the sliding rate of the nucleosome. Bulge-inchworm is another mechanism for intrinsic nucleosome sliding that we modeled. As a conclusion to the bulge model we argued that bulge pathway does not have an appreciative effect on the fate of site access. Our proposed models are more versatile compared to the available diffusion constant based models, firstly because we identify the individual micro events that comprise the process by their physical natures and characteristics, secondly the stochasticity of the event transitions would grant their proper weight in the process composite. The state of art of our models is that, whilst capturing the actual nature of biological events, they are accurate and computationally very fast. The proposed schema has important biological implications in explaining the *target site* access, in-vivo. Currently, we are working to model some of the prominent active pathways. Having these collection completed, we can give a comprehensive picture of pre-transcription events inside the nucleus in presence of TFs, Remodellers, and RNA II holoenzyme. Our model also links molecular properties of DNA and the location of the target sites on nucleosome component to the timing of transcription activation. This provides us with a general, predictive, parametric model for this biological function.

References

1. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter,*“Molecular Biology of the Cell”*, 4th Edition, Garland Science, 2002, ISBN 0815332181.
2. T. Richmond, and C. A. Davey, *“The Structure of DNA in Nucleosome Core”*, Nat. , vol 423, pp. 145-150, 2003.
3. G.u Li, M. Levitus, C. Bustamante, and J. Widom, *”Rapid spontaneous accessibility of nucleosomal DNA”*, Nat. Struc. & Mol. Bio., vol. 12, pp. 46-53, 2005.
4. D. A. Beard, and T. Schlick, *“Computational Modeling Predicts the Structure and Dynamics of Chromatin Fiber”*, Els. J. Strcu., Vol. 9, pp. 105114, 2001.
5. S. Ghosh, P. Ghosh, K. Basu, S. K. Das, and S. Daefler, *“iSimBioSys: A Discrete Event Simulation Platform for in silico study of biological systems”*, proc. of the 39th Annual Simulation Symp. (ANSS06), 2006.
6. P. Ghosh, S. Ghosh, K. Basu, and S. Das, *“A Stochastic model to estimate the time taken for Protein-Ligand Docking”*, IEEE CIBCB, 2006.

[3] In an-istorpic RW the probabilities of jumps to the left and right states are not necessarily equal.

7. K. Luger, A. W. Mader, A.W., R. K. Richmond, D. F. Sargent, and T. J. Richmond, *"Crystal structure of the nucleosome core particle at 2.8 resolution"*, Nature, vol. 389, pp.251260, 1997.
8. A. Flaus, T. Owen-Hughes, *"Mechanism for Nucleosome Mobilization"*, J. Biopolymers, vol 68, pp. 563-578, 2003.
9. J. Mellor, *"The Dynamics of Chromatin Remodeling at Promoters", J. Molecular Cell, vol 19, pp.147-157, 2005*
10. G. J. Narlikar, H. Fan, R. E. Kingston, *" Cooperation between Complexes that Regulate Chromatin Structure and Transcription"*, Cell, vol. 108, pp.475-487, 2002.
11. A. Saha, J. Wittmeyer, and B. R. Cairns, *"Cairns Chromatin remodelling: the industrial revolution of DNA around histones"*, Nat. rev. Mol. Cell Biol., vol. 7, pp. 437-447.
12. M. S. Kobor et al., *"A protein complex containing the conserved Swi2/Snf2-related ATPase Swr1p deposits histone variant H2A.Z into euchromatin"*, PLoS Biol. 2, E131, 2004.
13. X. Guo, K. Tatsuoka, and A. R. Liu, *"Histone acetylation and transcriptional regulation in the genome of Saccharomyces cerevisiae"*, Bioinformatics, vol. 22, pp. 392-399, 2006.
14. S. Pennings, G. Meersseman, and E. M. Bradbury, *"Mobility of positioned nucleosomes on 5 S rDNA"*, J. Mol. Biol., vol. 220(1), pp. 101-110, 1991.
15. B. Levin, *"Genes VIII"*, Pearson Prentice Hall , Upper Sadle River NJ, 2004, ISBN 0131439812.
16. K. Wiesenfeld and F. Jaramillo, *"Minireview of stochastic c resonance"*, Chaos, vol. 8, pp. 539-548, 1998.
17. S. Das, F. Sarkar, K. Basu, and S. Madhavapeddy,*"Parallel Discrete Event Simulation in Star Networks with Applications to Telecommunications*", Int. Workshop on Modeling, Analysis and Sim. of Computer and Telecom. Sys., 1995.
18. I.M. Kulic and H. Schiessel, *" Nucleosome repositioning via loop formation"*, Biophys. J., 2003.
19. Lumila V. Yakushevich, *"Nonlinear Physics of DNA"*, Wiley-VCH, Germany, 2004, ISBN:3527404171.
20. I.M. Kulic and H. Schiessel, *"chromatin Dynamics: Nuicleosome go Mobile through Twist Defects"*, Phy. rev. lett. , vol. 91(14), 2003.
21. Y. S. Kivshar, O. M. Braun, and J. S. Kivar, *"The Frenkel-Kontorova Model: Concepts, Methods, and Applications"*, Springer, 2004, ISBN:3540407715.
22. N. G. van Kampen *"Stochastic Processes in Physics and Cnimstrary"*, Elsevier pub., 1992, ISBN:0444893490.
23. Helmut Schiessel, *"The physics of chromatin"*, Max-Planck-Institut fur Polymerforschung, Theory Group, Mainz, Germany, 2003.
24. H. Schiessel, J. Widom, R. F. Bruinsma, and W. M. Gelbart, *"Polymer Reptation and Nucleosome Repositioning", Phy. rev. lett., vol.86(19), pp.4414-4417, 2001.*
25. L. Kleinrock, *"Queueing Systems, Vol. I: Theory "*, Wiley, New York, 1975.
26. P. Ghosh, S. Ghosh, K. Basu, S. Das and S. Daefler, *"An Analytical Model to Estimate the time taken for Cytoplasmic Reactions for Stochastic Simulation of Complex Biological Systems"*, IEEE Granular Computing Conf., 2006.
27. P. Ghosh, S. Ghosh, K. Basu, S. Das and S. Daefler, *"A Model to estimate the time taken for protein-DNA binding for Stochastic discrete event simulation of biological processes*, Accepted for pub. in IEEE CIBCB, USA, 2007.
28. A. R. Mazloom , K Basu and S. Das, *"A Random Walk Modelling Approach for Passive Metabolic Pathways in Gram-Negative Bacteria*", IEEE CIBCB, Canada, 2006.
29. K. P. N. Murthy and K. W. Kehr, *"Mean first-passage time of random walks on a random lattice*", Phys. rev A.,vol.(40),pp.2082, 1989.

Comparative Analysis of Gene-Coexpression Networks Across Species

Shiquan Wu and Jing Li*

Electrical Engineering and Computer Science Department
Case Western Reserve University, Cleveland, OH 44106, USA
{shiquan.wu,jingli}@case.edu

Abstract. This paper presents a large scale analysis of gene-coexpression networks (GCNs) across four plant species, i.e. Arabidopsis, Barley, Soybean, and Wheat, over 1471 DNA microarrays. We first identify a set of 5164 metagenes that are highly conserved across all of them. For each of the four species, a GCN is constructed by linking reliable coexpressed metagene pairs based on their expression profiles within each species. Similarly, an overall GCN for the four species is constructed based on gene expression profiles across the four species. On average, more than 50K correlation links have been generated for each of the five networks. A number of recent studies have shown that topological structures of GCNs and some other biological networks have some common characteristics, and GCNs across species may reveals conserved genetic modules that contain functionally related genes. But no studies on GCNs across crop species have been reported. In this study, we focus on the comparative analysis of statistical properties on the topological structure of the above five networks across Arabidopsis and three crop species. We show that: (1) the five networks are scale-free and their degree distributions follow the power law; (2) these networks have the small-world property; (3) these networks share very similar values for a variety of network parameters such as degree distributions, network diameters, cluster coefficients, and frequency distributions of correlation patterns (sub-graphs); (4) these networks are non-random and are stable; (5) cliques and clique-like subgraphs are overly present in these networks. Further analysis can be carried out to investigate conserved functional modules and regulatory pathways across the four species based on these networks. A web-based computing tool, available at *http://cbc.case.edu/coexp.html*, has been designed to visualize expression profiles of metagenes across the four species.

1 Introduction

With the availability of huge amount of genomic data, gene functions are usually predicted by similarity-based sequence analysis [7,9]. A great challenge in the post-genomic era is to understand gene regulations, genetic pathways and functional relations/modules of biological organisms at a system level [13,17,18,19,22].

[*] Corresponding author.

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 615–626, 2007.

For such a purpose, sequence-based analysis has its limitations because genes with even very similar sequences may not be functionally related to one another [11,19]. Therefore, it becomes essential to integrate both genomic information and microarray data (and some other data sources) in the discovery of gene regulatory and functional relations. However, it is still hard or even impossible to identify regulatory or functionally related genes if studies are limited to only a single species[19]. This motivates the investigation of gene regulatory and functional relations by integrating both genomic information and microarray data to study not only a single species, but across multiple species. Recently, much attention has been paid to the investigation of biological networks and/or conserved functional modules using multiple species, or multiple tissues. An earliest study has investigated gene-coexpression networks across humans, flies, worms, and yeast[19], and has discovered some global conserved genetic modules across these species. Since then, a number of studies[4,10,15] have been proposed to analyze complex gene-coexpression networks across species. Berg and Lässig[4] have proposed a Bayesian alignment method and have identified significant conservations of gene expression clusters and gene functions by analyzing GCNs between humans and mice. Lelandais et al. [15] have adopted the Multi-dimensional Scaling technique to compare GCNs from budding and fission yeasts and have extracted some common properties and difference between the two species. Guimera and Amaral[10] have proposed a method that can generate a 'cartographic representation' of biological networks which enables the identification of functional modules from those networks. They have applied the method on metabolic networks across twelve organisms and have discovered that nodes with different connectivity patterns are affected by different evolutionary constraints and pressures. Gene-coexpression networks across different tissues have also been studied by a number of groups [2,6,14] in order to identify conserved interactions among disease genes. Other researchers [3] have identified functionally related proteins by analyzing conserved protein-protein interactions across species.

These studies mainly focus on crossing humans, animals or diseases. None of them have investigated the properties of GCNs across plants. In this paper, we study the statistical properties [1] of gene coexpression networks across four plant species: Arabidopsis, Barley, Soybean, and Wheat using 1471 hybridizations, whose genomic and DNA microarray data are available at public webs. Arabidopsis is chosen here because it is a well-understood model organism and can be used to study the functionality of genes and/or functional modules in other three species, which are important crops in the world. To the authors' best knowledge, this is the first study on gene-coexpression networks of crop species together with a model organism Arabidopsis. It is of importance to understand the statistical properties of gene-coexpression networks in order to learn their functional relations, and to understand regulatory pathways among genes across these species. The current study on GCNs is an important step towards further understandings and studies of conserved functional modules from the four species.

The rest of this paper is organized as follows. In Section 2, we first introduce the data sources that include genomic sequences and 1471 DNA microarray expression profiles of the four plant species. A set of 5164 metagenes is then obtained by comparing the genes across the four species using BLAST. A web-based computing tool is designed to view the expressions of metagenes across various species, experiments, and hybridizations. Five gene-coexpression networks are then constructed based on the Pearson's correlation coefficient from the DNA microarray expression profiles of metagenes. We also propose a simple algorithm to calculate the frequency distribution of correlation patterns. In Section 3, a comparative analysis is conducted on the five gene-coexpression networks. We have obtained the following statistical properties for these networks: (1) the degree distributions of the five coexpression networks follow the power-law, *i.e.*, $P(k) \sim k^{-\gamma}$, which means the probability of a node with a degree of k ($P(k)$) is proportional to $k^{-\gamma}$, where $\gamma \geq 0$ is the exponent of the power-law [1]; (2) the five gene-coexpression networks have the small-world property [20], *i.e.*, the network diameters are small and the cluster coefficients are great; (3) the five gene-coexpression networks share very similar values across a variety of network parameters such as degree distributions, network diameters, cluster coefficients, and the frequency distributions of interaction/correlation patterns; (4) these networks are non-random and their properties are stable under randomly introduced noise, even with as large as 20% changes of edges; (5) cliques and clique-like subgraphs are overly present in these gene-coexpression networks. In Section 4, we conclude the paper by discussing some potential work in predicting functional modules and regulatory pathways using these coexpression networks from multiple species.

2 Materials and Methods

2.1 Sequence and Expression Data

The materials used in this study include the genomic sequences and a large set of DNA microarray expression profiles of the four plant species, which were collected from several public sources on the Internet: *http://affymetrix.com, http://arabidopsis.org, http://tigr.org, http://ausubellab.mgh.harvard.edu/imds, http://psi081.ba.ars.usda.gov/SGMD, http://soybeangenome.org, http://harvest-web.org, http://plexdb. org, http://www.ncbi.nlm.nih.gov/projects/geo, http:// smd.stanford.edu, etc.* For Arabidopsis, we select 617 DNA microarray expression profiles. For Barley, Soybean, and Wheat, we respectively have 671, 53, and 130 microarrays. These 1471 DNA microarray expression profiles contain diverse conditions of microarray experiments (*e.g.*, various experimental organisms, different experimental types, a wide range of experimental factors, *etc.*)

The aim of this study is to investigate the common orthologous genes of the four species and the statistical properties of the gene-coexpressions across the species, experiments, and hybridizations.

There are three major steps in this study: (1) identifying metagenes across the four species; (2) constructing five gene-coexpression networks based on microarray expression profiles of the metagenes: one for each of the four individual species, and one for the overall gene-coexpressions across the four species; (3) investigating the statistical properties of the five gene-coexpression networks by a comparative analysis.

2.2 Identifying Metagenes

By applying "all-against-all" BLAST[19] to all genes of each pair of species, 5146 metagenes are obtained and shown in Table 1 (a complete list of the 5164 metagenes is available on our website). These 5164 metagenes are only a small fraction of all the genes with expression data from each species. Each metagene is defined as a set of four genes, one from each of the four species. Any two genes in a metagene are each other the best hit by BLAST using their protein sequences. For example, Metagene 1 consists of four genes: "244901_at" in Arabidopsis, "Barley1_53087" in Barley, "GmaAffx.25198.1.S1_at" in Soybean, and "Ta.22468.1.S1_at" in Wheat (see the first row in Table 1). These four genes are the best hits each other by BLAST using their protein sequences. Metagenes setup a mapping between the genes of one species and those of another species. By this mapping, it is possible to analyze gene expressions across various species, experiments, and hybridizations. The expressions of metagenes across species, experiments, and hybridizations can be viewed by our web-based computing tool at *http://cbc.case.edu/coexp.html*.

Table 1. Metagenes across Arabdopsis, Barley, Soybean, and Wheat

No.	**Arabidopsis**	**Barley**	**Soybean**	**Wheat**
1	244901_at	Barley1_53087	GmaAffx.25198.1.S1_at	Ta.22468.1.S1_at
2	244936_at	Barley1_53095	GmaAffx.17247.1.S1_at	TaAffx.124056.1.S1_at
···	··· ···	··· ···	··· ···	··· ···
5164	267646_at	Barley1_07944	Gma.3262.1.S1_at	Ta.10084.1.S1_at

2.3 Constructing Gene-Coexpression Networks

Gene-coexpression networks are constructed based upon metagenes' DNA microarray expression profiles. Relabel the 1471 hybridizations and denote H_1, H_2, $\cdots$, and H_{617} the 617 hybridizations of Arabidopsis; H_{618}, H_{619}, $\cdots$, and H_{1288} the 671 hybridizations of Barley; H_{1289}, H_{1290}, $\cdots$, and H_{1418} the 130 hybridizations of Soybean; and H_{1419}, H_{1420}, $\cdots$, H_{1471} the 53 hybridizations of Wheat. These 1471 DNA microarrays define the following 5164 $\times$ 1471 matrix of intensities, where the k_{th} row of the matrix represents the expression intensities of the k_{th} metagene across the 1471 hybridizations. Whereas each column represents the expression intensities of all metagenes under a hybridization. The hybridizations within each experiment have been normalized when we downloaded the data. However, when the hybridizations from various experiments and different species are put together, a new normalization is needed. The expression intensities are normalized across species and experiments using the Quantile normalization method[5]. The purpose is to adjust the effects arising from variation of different experiments rather than from biological differences [5,21].

To construct a gene-coexpression network $G = (V, E)$, we take each metagene as a node in V. For each pair of metagenes: k and j, the Pearson's correlation coefficient $(r(k, j))$ based on their expression profiles can be calculated as $r(k, j) = \frac{\Sigma KL - \frac{\Sigma K}{\Sigma L}}{\sqrt{(\Sigma K^2 - \frac{(\Sigma K)^2}{N})(\Sigma L^2 - \frac{(\Sigma L)^2}{N})}}$, where K and L represent the k_{th} and l_{th} row vectors of the intensity matrix, and N is the number of hybridizations. If $|r|$ is greater than a predefined cutoff value (the choice of the exact value of the threshold will be discussed below), the expressions of metagenes k and j are highly correlated and an edge is added between the pair. When constructing a gene-coexpression network, a proper cutoff value is necessary so that only significant correlations are included in a coexpression network. In this study, a similar approach as in[14] has been used in determining threshold values for different datasets. Basically, under the null hypothesis of no correlation, the Pearson correlation coefficient corresponds to a t-distribution with degrees of $N - 2$. An overall error rate of 0.05 is chosen after Bonferroni correction of multiple testing. In addition, only top and bottom 0.5% of correlations will be included for further study[14]. The combination of criteria corresponds to cutoff values from 0.8 to 0.9 in this study.

There are five gene-coexpression networks being constructed as follows. For Arabidopsis, the gene-coexpression network $G_{AT} = (V, E_{AT})$ is constructed based on the microarray data of Arabidopsis: H_1, H_2, $\cdots$, and H_{617}, with a cutoff value 0.8. Similarly, for Barley, Soybean, and Wheat, their gene coexpression networks $G_{BB} = (V,\ E_{BB})$, $G_{GM} = (V, E_{GM})$, $G_{TA} = (V, E_{TA})$ are respectively constructed by using H_{618}, H_{619}, $\cdots$,H_{1288} with cutoff value 0.8 for G_{BB}; H_{1289}, H_{1290}, $\cdots$, H_{1418} with cutoff value 0.85 for G_{GM}; H_{1419}, H_{1420}, $\cdots$, $H_{1471}$with cutoff value 0.9 for G_{TA}. Finally, $G = (V, E)$ is constructed as an overall gene coexpression network across four species by using all 1471 DNA microarrays with cutoff value 0.8. Therefore, G_{AT}, G_{BB}, G_{GM} and G_{TA} respectively represent metagene pairs that are coexpressed with significant correlations within the experiments of each individual species. Whereas, G represents metagene pairs coexpressed with significant correlations in all experiments across the four species. Each of the networks has 5164 nodes and the number of edges is in the range of 50k-70k (see Table 2). A possible explanation that different networks have different cutoff values is that the numbers of hybridizations for different networks vary dramatically, from around 600 for G_{AT} and G_{BB} to 100 for G_{GM} and 50 for G_{TA}. Greater cutoff values for smaller data sets are chosen to ensure only significant correlations being included.

2.4 Statistical Analysis of Network Parameters

The aim of this study is to investigate the statistical properties of network parameters that include degree distributions, network diameters, and clustering coefficients from the five gene-coexpression networks. These parameters have been widely discussed for large-scale complex networks and the calculation can be performed based on their definitions [1].

In addition to expression links, some subgraphs in gene-coexpression networks may represent important functional modules, and it is of great interest to understand or identify special patterns of subgraphs that are overly represented from GCNs across species. As the first step, we have studied the frequency distribution of correlation patterns/subgraphs in this paper that have similarly been taken into considerations by other researchers [19]. In total, 29 correlation patterns/subgraphs each with 3 to 5 nodes have been included (see Fig. 2). For each of the patterns, its frequency in a network is defined as the number of its occurrences in the network. If a substructure has been counted as one pattern (say Pattern 17 in Fig. 2), none of its subgraphs with the same number of nodes (say Pattern 12 and 16) will be counted as occurrences of smaller patterns. A simple computer program has been implemented to count the frequencies of the 29 correlation patterns in a network based on the following algorithm. The running time of the algorithm depends on the degree distribution of a network. But it is much faster than the naive method that examines every subset with 5 nodes.

Algorithm: Counting Pattern Frequency

```
 Input  Network $G = (V, E)$ (denote $V$ by integers: 1,2,. . .,5164).
Output  Frequency $f_k$ (denote $P_k$ the $k_{th}$ pattern, $1 \leq k \leq 29$).

Step 1  For each node $i$, find its neighbors:
        $N_i = \{j|(j, i) \in E, j > i\}, 1 \leq i \leq 5164$.

Step 2  For each $N_i$, check every subset $U$ of $N_i$ with $|U| \leq 4$,
        if $\{i\} \cup U$ is a pattern, say $P_k = G_{\{i\} \cup U}$, and $G_{\{i\} \cup U}$
        is not contained in any other patterns with $|U| + 1$ nodes,
        count the occurrence into the frequency of $P_k$, i.e. $f_k$++.
```

3 Results

In this section, we present the statistical properties of network parameters obtained by the comparative analysis of the five gene-coexpression networks. First, a brief summary on the network parameters of the five networks is given in Table 2. It can be seen that all the parameters are very similar across the five gene coexpression networks. This is probably because the four plant species have relatively small evolutionary distances from each other.

As observed in many gene expression networks by other researchers, the GCNs obtained here also have the small-world property, *i.e.*, they all have small network diameters (either 4 or 5) and great cluster coefficients (at least 0.6). The degrees of the five GCNs follow the power-law distributions with very similar power-law exponents γ (Table 2). The values of γ (1.13-1.28) are consistent with many other gene-coexpression networks obtained by other researchers(see [2] and references therein). The degree distributions of the five GCN are displayed

Table 2. Summary of network parameters

Network	Node	Edge	Power-law exponent	Network diameter	Cluster coefficient
G_{AT}	5164	56392	1.1520	5	0.7042
G_{BB}	5164	52714	1.2147	4	0.8264
G_{GM}	5164	72968	1.1298	4	0.7827
G_{TA}	5164	63382	1.1569	4	0.6338
G	5164	51905	1.2777	4	0.6444

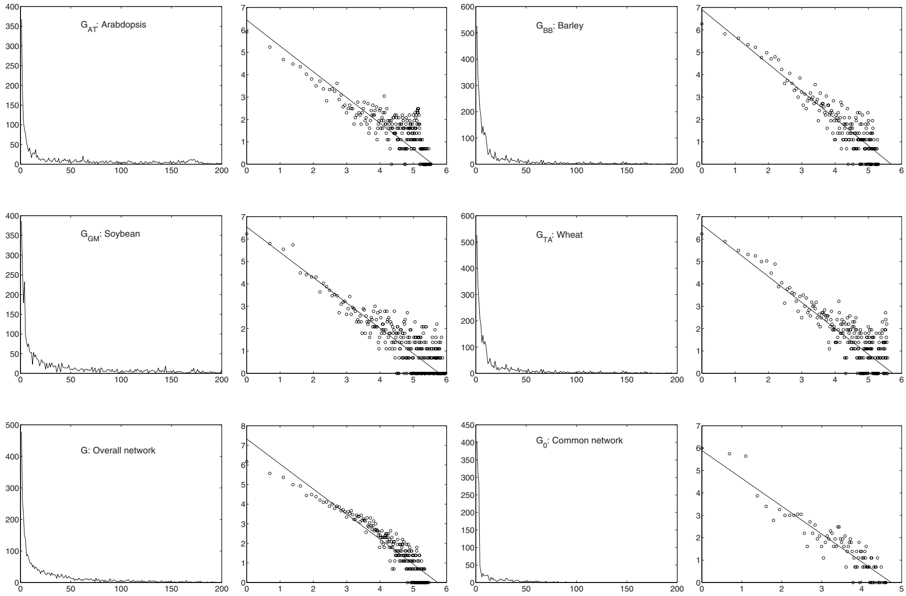


Fig. 1. The first and third columns are the degree distributions of G_{AT}, G_{BB}, G_{GM}, G_{TA}, G and G_0, respectively, where $x-$axis represents the number of degrees and $y-$axis represents the number of nodes. The second and forth columns are the log-log plots of the degree distributions of $G_{AT}, G_{BB}, G_{GM}, G_{TA}, G, G_0$.

in Fig. 1, together with their log-log plots. In addition, we have defined a new network $G_0 = (V, E_0)$ (called the common network) by taking all the common edges from the four gene-coexpression networks for individual species, i.e. $E_0 = E_{AT} \cap E_{BB} \cap E_{GM} \cap E_{TA}$. The degrees of G_0 also follows a power-law distribution, shown in Fig. 1, but the total number of edges in G_0 is much smaller compared with other five networks.

Distributions of correlation patterns/subgraphs. Genes and proteins always interact with each other in groups to perform certain biological functions. It is important to understand their interaction/correlation patterns. As the first step, we evaluate the frequency distributions of 29 correlation patterns (also see [16], each of which has 3 to 5 nodes) in the gene-coexpresion networks obtained in this study. Each pattern may represent a particular type of the interactions/correlations of the genes/metagenes. For comparison purpose, we further introduce two new networks. The perturbed network under noise G_S is defined by introducing as much as 20% noise on edges, *i.e.*, deleting 10% of the existing edges and adding 10% of new edges in the network G. And a random network G_R is generated, which consists of the same number of nodes (5164). An edge will be added to each pair of nodes in G_R with a small probability so that the total number of edges in G_R will be similar as the number of edges in other networks. The frequency distributions of the 29 patterns in all eight networks, *i.e.*, $G_{AT}, G_{BB}, G_{GM}, G_{TA}, G, G_S$, G_0 and G_R, are counted using the algorithm described in subsection 2.4 and shown in Fig. 2.

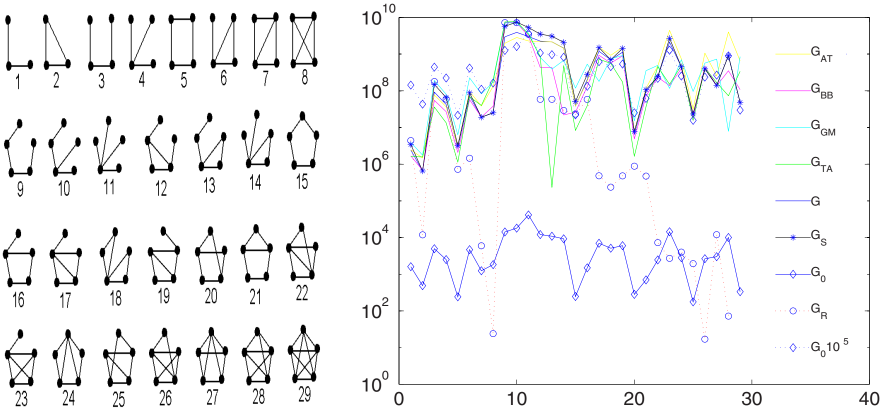


Fig. 2. Left: the 29 correlation patterns (also see [16]). Right: the frequency distributions of the 29 patterns; Top part: $G_{AT}, G_{BB}, G_{GM}, G_{TA}, G, G_S$ (presented by "$*$"), $G_0 \cdot 10^5$(presented by "$\diamond$"); Bottom part: G_0 (presented by "$\diamond$"); G_R is indicated by "$\circ$": up and down between top and bottom. $x-$axis: 29 patterns. $y-$axis: frequency.

The five gene-coexpression networks ($G_{AT}, G_{BB}, G_{GM}, G_{TA}, G$), as well as the perturbed network G_S, have very similar frequency distributions over all the 29 patterns. The frequencies in graph G_0 are much lower because G_0 consists of a much smaller number of edges. But the distribution pattern is the same as those of other coexpression networks. To make it clear, we multiple the frequency distribution in G_0 by 10^5 and denote the new scaled distribution by $G_0 \cdot 10^5$. Fig. 2 shows that all the 7 gene coexpression networks have very similar distributions across all the patterns, but the random network G_R has a very different distribution. Further examinations reveal that a subset patterns (such as patterns 2, 7-8, 22-29) in the 7 coexpression networks have very different frequencies from the random network G_R. Those patterns are either cliques (patterns 2, 8 and 29) or some condensed patterns that are very similar to cliques (patter 7 and patterns 22-28). The over presence of cliques or clique-like subgraphs in GCNs may reflect the facts that those genes in a clique may encode proteins that form a protein complex, or they may be regulated by a common transcription factor. More investigations are needed on these overly presented patterns.

In order to quantitatively measure the overall differences of frequency distributions of the 29 patterns among G_{AT},G_{BB},G_{GM},G_{TA}, G, G_S, and G_R, a distance measure is defined as $d(G_i, G_j) = \sum_{k=1}^{29} w_k |log(n_{ik}) - log(n_{jk})|$, where n_{ik} and n_{jk} are the frequencies of Pattern k of G_i and G_j, respectively, and w_k is the weight of Pattern k ($w_k \geq 0$ and $\sum_{k=1}^{29} w_k = 1$). In this study, we simply take an equal weight for each pattern, *i.e.*, $w_k = 1/29$ for any k. The pairwise distances among all the networks are given in Table 3. The distances show that G_{AT},G_{BB},G_{GM},G_{TA}, G, and G_S (also $G_0 \cdot 10^5$) are very close each other (all pairwise distances of the 6 networks are less than 1.66). In contrast, the random network G_R is quite different from them (the distance between G_R and any other one is around 6, see the last row).

Table 3. Distances on frequency distributions

Distance	G_{AT}	G_{BB}	G_{GM}	G_{TA}	G	G_0	$G_0 \cdot 10^5$	G_S	G_R
G_{AT}	0	0.99	1.07	1.30	0.49	11.39	1.19	0.61	6.81
G_{BB}	0.99	0	1.11	0.93	0.63	10.57	1.33	0.76	5.91
G_{GM}	1.07	1.11	0	1.27	1.21	11.29	1.35	1.14	6.41
G_{TA}	1.30	0.93	1.27	0	1.26	10.32	1.66	1.38	6.18
G_G	0.49	0.63	1.21	1.26	0	11.00	1.06	0.22	6.33
$G_0 \cdot 10^5$	1.19	1.33	1.35	1.66	1.06	11.40	0	1.14	6.73
G_S	0.61	0.76	1.14	1.38	0.22	11.22	1.14	0	6.37
G_R	6.81	5.91	6.41	6.18	6.33	19.13	6.73	6.37	0

Non-randomness and stability of coexpression networks. We believe that the correlation links and the overly presented patterns in the networks are statistically significant and they might be biologically meaningful. First of all, the analysis was performed on a large data set that consists of 1471 hybridizations. It is unlikely to obtain significant correlations and expression patterns by chance over such a large data set. Furthermore, we have constructed a random network G_R and a perturbed network G_S. The network parameters of the two networks are shown in Table 4 and their degree distributions are shown in Fig. 3. It is obvious that the gene-coexpression networks are quite different from the random network in terms of degree distributions, cluster coefficients, and pattern frequency distributions. On the other hand, all the parameters from the perturbed network are very similar with those from all other gene-coexpression networks. This indicates that the results obtained from this study are quite robust and can not be generated by chance.

Biological meaning of the gene coexpression networks. The networks we construct represent significant correlations among metagenes across the species over a large set of microarray data. Various types of subgraphs in these coexpression networks may imply biological meaningful properties or functional relations of genes. We first take the top three hub nodes from the Arabidopsis network

Table 4. Network parameters of G_R and G_S

Network	Node	Edge	Power-law exponent	Network diameter	Cluster coefficient
G_R	5164	53461	Non Power-law	4	0.008
G_S	5164	73119	1.2672	6	0.604

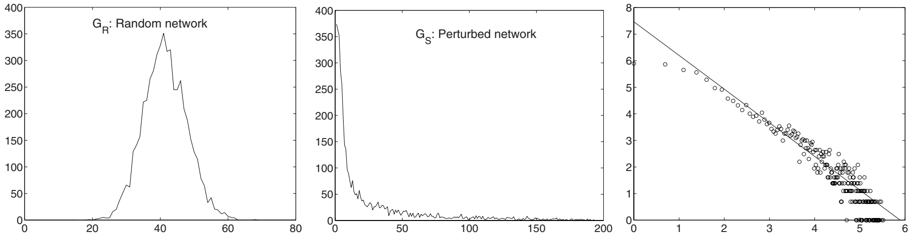


Fig. 3. Left: the non-power-law degree distribution of G_R. $x-$axis: number of degrees; $y-$axis: number of nodes. Middle: the degree distribution of G_S. Right: the log-log plot of the degree distribution of G_S.

and check their GO (Gene Ontology) annotations(*http://www.arabidopsis.org*). We find that the genes represented by the hub nodes are involved in protein expressing, folding, and binding, which are essential in protein synthesis and protein-protein interactions. The corresponding metagenes also have large number of links in the coexpression networks of other three species (due to the page limitation, details of the results in this subsection can be found from our website). The above observations suggest that our coexpression networks may reveal certain important biological properties. Gene functions from Barley, Soybean, and Wheat, which are mostly unknown, can be predicted through our coexpression networks and functional annotations of Arabidopsis.

To further explore the networks, we choose those highly significant links by a cut-off value of 0.99. The node with the largest number of links in Arabidopsis (gene 246075_at) has 47 neighbors. This gene has its GO annotation as transferase activity, transferring glycosyl groups, and UDP-galactosyltransferase activity. Among all the 48 genes, more than 70% of all the gene pairs (48 choose 2) are linked. Most genes in this subgraph share similar GO annotations such as catalytic activity, cellulose synthase activity, transferase activity, and kinase activity. Therefore this subgraph may present one or several groups of functionally related genes. A few genes with unknown biological functions in this subgraph may be predicted based on annotations of other genes within the same group.

4 Discussions

In this study, we first obtained metagenes that are orthologous genes in common to the four plant species by sequence analysis. Those metagenes might have been conserved from their common ancestor through evolution and might play an important role in biological functions and regulations. Gene-coexpression networks were then constructed based on their expression profiles. We have investigated the statistical properties of those gene-coexpression networks. The degrees of all gene-coexpression networks follow power-law distributions and they have the small-world property with small network diameters and great cluster coefficients. The values of those parameters from all the expression networks are very similar, probably because the four species are very close in evolution. The properties are quite different from those of a random network and are robust under perturbation. We have also investigated the frequency distributions of 29 correlation patterns and have found that cliques and clique-like patterns are overly present in these networks but not in a random network with similar size. This result implies that some of those patterns may represent certain important functional modules. Further studies are needed to explore the biological meanings of those patterns.

This study has two significant features. First, it is the first study on the gene-coexpression networks across crop species, whereas previous studies mainly focus on the gene-coexpression networks of single crop species[8], or across humans, animals and diseases[2,3,4,6,10,14,15,19]. Secondly, this study first investigates the statistical properties of the frequency distributions of correlation patterns in gene-coexpression networks and has identified that cliques and clique-like patterns are overly present in these networks. Previous studies mainly discuss network parameters such as degree distributions, diameters, cluster coefficients.

Pathways and gene regulatory networks are usually predicted by comparative genomics using sequence information, which can also be applied on crops such as Barley, Soybean, and Wheat. However, metagenes and coexpression networks can be used as a new method to predicting functionally related genes, functional modules and regulatory pathways. By across-species inference, the known functionally related genes, functional modules and regulatory pathways of one species can be used to predicting those of other species. Arabidopsis is a well-studied species. Many functionally related genes, functional modules and regulatory pathways have been identified in Arabidopsis. Whereas, little has been known on the pathways, regulations, functions, and modules about Barley, Soybean, and Wheat. Gene-coexpression networks can make it possible to predict functionally related genes, functional modules and regulatory pathways in the three species by those in Arabidopsis. If a group of coexpressed metagenes are functionally related in Arabidopsis, by comparing gene-coexpression networks, it is possible to predict that those metagenes may also be functionally related in Barley, Soybean, and Wheat. We will address this issue in our future studies.

Acknowledgements. This research was supported in part by NIH/NLM grant LM008991 and a start-up fund from Case Western Reserve University to JL.

References

1. Albert, B. and Barabási, A.-L.: Statistical mechanics of complex networks, *Review of Modern Physics*, 74(2002)47-97.
2. Aggarwal, A., Guo, D. L., etc.: Topological and Functional Discovery in a Gene Coexpression Meta-Network of Gastric Cancer, *Cancer Research*, 16(2006)232-241.
3. Bandyopadhyay, S., Sharan, R. and Ideker, T.: Systematic identification of functional orthologs based on protein network comparison, *Genome Research*, 16(2006)428-435.
4. Berg, J. and Lässig, M.: Cross-species analysis of biological networks by Bayesian alignment, *PNAS*, 103(2006)10967-10972.
5. Bolstad, B. M., Irizarry, R. A., Astrand, M. and Speed, T. P.: A Comparison of Normalization Methods for High Density Oligonucleotide Array Data Based on Bias and Variance, *Bioinformatics*, 19(2003)185-193.
6. Choi, J. K., Yu, U., Yoo, O. J. and Kim, S.: Differential coexpression analysis using microarray data and its application to human cancer, *Bioinformatics*, 21(2005)4348-4355.
7. Durbin, R., Eddy, S. R., Krogh, A. and Mitchison, G.: *Biological sequence analysis: Probabilistic models of proteins and nucleic acids*, Cambridge Univ. Press (1998).
8. Faccioli, P., Provero, P., etc: From single genes to co-expression networks: extracting knowledge from barley functional genomics, *Plant Molecular Biology*, 58(2005)739-750.
9. Gusfield,D.: *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, Cambridge University Press (1997).
10. Guimera, R. and Amaral, L. A. N.: Functional cartography of complex metabolic networks, *Nature*, 443(2005)895-900.
11. Gerlt, J. A. and Babbitt, P. C.: Can sequence determine function?, *Genome Biology*, 1(2000):REVIEWS0005.
12. Hanfrey, C., Sommer, S., etc.: Arabidopsis polyamine biosynthesis: absence of ornithine decarboxylase and the mechanism of arginine decarboxylase activity, *Plant Journal*, 27(2001)551-60.
13. Kitano, H.: Systems Biology: A Brief Overview, *Science* 295(2002)1662-1664.
14. Lee, H. K., Hsu, A.K., etc.: Coexpression analysis of human genes across many microarray data sets, *Genome Research*, 14(2004)1085-1094.
15. Lelandais, G., Vincens, etc.: Comparing gene expression networks in a multidimensional space to extract similarities and differences between organisms, *Bioinformatics*, 22(2006)1359-1366.
16. Pržulj, N., Corneil, D. G. and Jurisica, I.: Modeling interactome: scale-free or geometric? *Bioinformatics*, 20(2004)3508-3515.
17. Sali, S.: Functional links between proteins, *Nature*, 402(1999)23-26.
18. Strogatz, S. H.: Exploring complex networks, *Nature*, 410(2001)268-276.
19. Stuart, J. M., Segal, E., Koller, D., and Kim, S. K.: A Gene-Coexpression Network for Global Discovery of Conserved Genetic Modules, *Science*, 302(2003)249-255.
20. Watts, D. J. and Strogatz, H.: Collective dynamics of 'small-world' networks, *Nature*, 393(1998)440-442.
21. Yang, Y. H., Dudoit, S., etc.: Normalization for cDNA microarray data: a robust composite method addressing single and multiple slide systematic variation, *Nucleic Acids Research*, 30N4(2002)e15.
22. Zhou, X. J. and Gibson, G.: Cross-species comparison of genome-wide expression patterns, *Genome Biology*, 5(2004)232-233.

Comparative Pathway Prediction Via Unified Graph Modeling of Genomic Structure Information

Jizhen Zhao, Dongsheng Che, and Liming Cai

Department of Computer Science, University of Georgia, Athens, GA 30602, USA
Fax: (706) 542-2966
{jizhen,che,cai}@cs.uga.edu

Abstract. Genomic information other than sequence similarity is important for comparative analysis based prediction of biological pathways. There is evidence that structure information like protein-DNA interactions and operons is very useful in improving the pathway prediction accuracy. This paper introduces a graph model that can unify the protein-DNA interaction and operon information as well as homologous relationships between involved genes. Under this model, pathway prediction corresponds to finding the maximum independent set in the model graph, which is solved efficiently via non-trivial tree decomposition-based techniques. The developed algorithm is evaluated based on the prediction of 30 pathways in *E. coli* K12 using those in *B. subtilis* 168 as templates. The overall accuracy of the new method outperforms those based solely on sequence similarity or using different categories of structure information separately.

Keywords: Pathway prediction, protein-DNA interaction, operon, tree decomposition, independent set, clique.

1 Introduction

It is important yet challenging to understand the roles of genes and their relationships for many sequenced genomes [14]. In addition to the analysis of individual gene functions, the task of pathway annotation, assigning a biological pathway to a set of genes, plays a fundamental role in the higher order functional analysis of organisms and in the understanding of cellular processes and organism behaviors in a larger context [16]. Experimentally determining biological pathways is usually expensive and laborious. Computational methods based on genomic information to predict pathway are more desirable. In particular, it is feasible to employ comparative genomic analysis in pathway prediction at the genome scale. Based on an annotated pathway from one genome as template, a pathway for a target genome can be predicted by identifying a set of orthologues based on sequence similarity to the genes in the template pathway. A naive approach for orthology assigning is choosing the best BLAST hit for each gene (BH). A more often used technique is by reciprocal BLAST search, called bidirectional best-hit

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 627–637, 2007.

(BBH) [10], where gene pairs are regarded as orthologues if they are the best hits in both directions of the search. However, these and other sequence similarity based approaches share the same limitation [9]: the best hits may not necessarily be the optimal orthologues, thus compromising the prediction accuracy.

Homology relationships between genes can exist not only at the sequence level, but also at functional and structural levels [17], especially those of operon structures and transcriptional regulation patterns of genes by transcriptional factors (TFs). Such high level categories of information among genes along with the sequence similarity can be used to improve the pathway prediction accuracy. Without doubt, research in the collection of genomic structure information has appeared to support such an endeavor as well. For example, recently, substantial operon and transcriptional regulation information have been curated from the scientific literatures for a number of genomes [8,13]. Computational methods [12,13,17] have also been developed to predict operon structures and co-regulated genes.

It is computationally challenging to incorporate genomic structure information into others for the pathway prediction. The optimal prediction of pathways at the genome scale becomes intractable combinatorial optimization problems if sophisticated structure information is considered. PMAP [9] is an existing method that addresses this issue by incorporating partial structure information (i.e. structure information of the target genome only) and solving it with integer programing.

In this paper, we introduce a unified graph model for integrating data in sequence similarity, gene functions, experimentally confirmed or predicted operons, and transcriptional regulations, of both template pathway and target genomes. The new approach has led to an efficient pathway prediction algorithm TdP that outperforms the existing ones in prediction accuracy.

Algorithm TdP predicts a pathway in a target genome based on a template pathway by identifying an orthologous gene in the target genome for each gene in the template pathway, such that the overall sequence and structural similarity between the template and the predicted pathways achieves the optimal. In particular, homologes for each gene in the template pathway are first identified by the BLAST search [1]. Functional information is then used to filter out genes unlikely to be orthologues. The structural information are used to constrain the orthology assignment. One of the homologes is eventually chosen to be the ortholog for the gene. The pathway prediction is formulated into the maximum independent set (MIS) problem by taking protein-DNA interaction and operon structures all together. Because problem MIS is computationally intractable, we solve them efficiently with non-trivial techniques based on tree decompositions of the graphs constructed from the structural information.

Our algorithm TdP has been implemented and its effectiveness is evaluated against BH, BBH and PMAP in the annotation of 30 pathways of *E. coli* K12 using the corresponding pathways of *B. subtilis* 168 as templates. The results showed that overall, in terms of the accuracy of the prediction, TdP outperforms BH and BBH that are based solely on sequence similarity, as well as PMAP that uses partial structural information. In term of average running time to predict a pathway, it outperforms PMAP. The core of algorithm TdP is dynamic programing based on tree decomposition techniques. The running time of the dynamic programing is dominated by function $2^t n$, where t is the tree width of the input graphs for dynamic programming of n vertices constructed from the structural information. In particular, the statistics on the tree width of these graphs shows that about 93% of the graphs have tree width at most 5. Therefore, the tree decomposition based algorithm for pathway prediction is more efficient theoretically and practically than the existing algorithm PMAP based on integer programming.

2 Methods and Algorithm

2.1 Problem Formulation

A *pathway* is defined as a set of molecules (genes, RNAs, proteins, or small molecules) connected by links that represent physical or functional interactions. It can be reduced to a set of genes that code related functional proteins. An *operon* is a set of genes transcribed under the control of an operator gene. Genes that encode transcriptional factors are called *tf* genes. In the work described in this paper, a known pathway in one genome is used as a template to predict a similar pathway in a target genome. That is, for every gene in the template pathway, we identify a gene in target genome as its ortholog if there is one, under the constraints of protein-DNA interaction (*i.e.*, transcriptional regulation) and operon information. The problem of predicting pathways is defined as:

Input: a template pathway model $P = \langle A_P, R_P, O_P \rangle$ and a target genome T, where A_P is a set of genes in P, R_P is a set of relationships between *tf* genes and genes regulated by corresponding *tf* gene products, and O_P is a set of operons;
Output: a pathway $Q = \langle A_Q, R_Q, O_Q \rangle$ for T and an orthology mapping $\pi : A_Q \rightarrow A_P$ such that the score function

$$s_1(A_Q, A_P, \pi) + s_2(R_Q, R_P, \pi) + s_3(O_Q, O_P, \pi) \qquad (1)$$

is the maximum.

In the above definition, the predicted pathway Q for the target genome T is quantified with three functions: function s_1 measures the sequence similarity between all genes in pathway Q and their corresponding orthologues in the template P; functions s_2 and s_3 measure the regulation consistency and operon consistency between pathways P and Q respectively.

2.2 The Methods

Our approach TdP consists of the following steps:

1. For every gene in the template pathway P, find a set of homologes in the target genome T with BLAST;

2. Remove from the homologous genes unlikely to be orthologues to the corresponding gene in the template P. This is done based on functional information, e.g., Cluster of Orthologous Groups (COG)[18], which is available. In particular, genes that are unlikely orthologs would have different COG numbers.
3. Obtain protein-DNA interactions and operon structures for the genes in the template pathway and the homologous genes in target genome from related databases [8,13], literatures or computational tools [12,17]. Although the more complete information the better, partially available information could also be used.
4. For each template gene not covered by the available structural information, one of its homologs is assigned as its ortholog if they are among the top three hits in both directions of BLAST search. We need to point out that we arbitrary choose top three here to relax bidirectional best hit constraint that may not be true in reality. Alternative finer strategies could be used here.
5. For template genes covered by the available structural information, exactly one of the homologous genes is eventually assigned as the ortholog for the corresponding gene in the template. This is done by a procedure *mapping*, which seeks the genes in the target genome with overall optimal sequence similarity and structural consistency to the corresponding genes in the template.

This method optimally incorporate the structural information, i.e. the regulation and operon information, into sequence similarity in the procedure *mapping*. Thus, such an orthology mapping or assignment after these steps essentially yield a predicted pathway that has overall optimal sequence similarity and structural consistency with the template pathway.

By incorporating sophisticated unified structural information, the orthology mapping constrained by the structural information mentioned in step (5) may become computationally intractable. We describe in the following in detail how the efficient procedure *mapping* can be obtained.

Let genes be vertices and relations among the genes be edges, the genes with available structural information (*i.e.* the transcriptional regulation and operon information) in the template pathway and the corresponding homologs in target genome can be naturally formed into two graphs. The optimal common subgraph of these two graphs corresponds to the optimal orthology assignment respecting to sequence similarity and structural consistency for involved genes. This is done by a procedure called *OptimalOrth*, which takes the involved genes in the template pathway, the corresponding homologs in target genome, together with the related structural information as inputs, and output the optimal orthology assignment for a sub set of the input template genes. Since one call of the procedure *OptimalOrth* may not assign orthologs for all of the input template genes, the procedure *mapping* may need to call *OptimalOrth* on reduced gene set of the template pathway and related homologs in target genome with related structural information. Next we first describe the procedure *OptimalOrth*, then give the details of the procedure *mapping*.

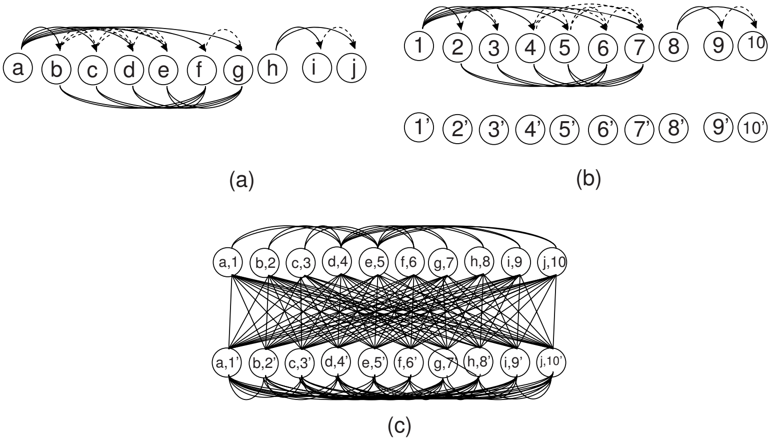


Fig. 1. Orthology mapping with structural information. (a) Structure graph G_1 for template pathway. A directed edge points from a *tf* gene to a gene regulated by the corresponding TF, a dashed edge connects two genes belonging to a same operon, a solid edge connects two genes regulated by a same TF but belonging to different operons. (b) Structure graph G_2 for the homologous genes in the target genome, constructed in similar way to (a). (c) Merged graph G from G_1 and G_2. Each node is a pair of homologous genes.

Procedure *OptimalOrth***.** The inputs of the procedure *OptimalOrth* are g_1, a set of genes with structural information in one genome, and g_2, the related homologs of g_1 with the structural information among them in another genome. It outputs the optimal orthology assignment for a sub set of g_{tem} in a set *orth*, which indeed are a number of ortholog pairs (i, j) where $i \in g_1$ and $j \in g_2$. It includes the following steps:

1. A structure graph $G_1 = (V_1, E_1)$ is built for the gene set g_1, where vertex set V_1 represents all genes in it, and edge set E_1 contains three types of edges: an edge of type-1 connects a *tf* gene and every gene regulated by its corresponding product; an edge of type-2 connects two genes belonging to a same operon; and edges of type-3 connect two genes regulated by the same *tf* gene product but belonging to different operons (Figure 1(a)).
2. A structure graph $G_2 = (V_2, E_2)$ is built for the homologous gene set g_2 in the similar way, where V_2 represents homologous genes in it (Figure 1(b)).
3. Graphs G_1 and G_2 are merged into a single graph $G = (V, E)$ such that V contains vertex $[i, j]$ if and only if $i \in V_1$ and $j \in V_2$ are two homologous genes. A weight is assigned to vertex $[i, j]$ according to the BLAST score between genes i and j. Add an edge $([i, j], [i', j'])$ if either (a) $i = i'$ or $j = j'$ but not both, or (b) $i \neq i'$ and $j \neq j'$ and either one of (i, i') and (j, j') is an edge but not both or edges $(i, i') \in E_1$ and $(j, j') \in E_2$ are not of the same type (Figure 1(c)).
4. Then the independent set in the graph G with the maximum weight should correspond to the optimal common subgraph of graph G_1 and G_2, which in

turn corresponds to desired orthology mapping that achieves the maximum sequence similarity and structure consistency between the involved genes in g_1 and the ones in g_2. The maximum weighted independent set is identified by a tree decomposition based procedure *mis*, which will be described in section 2.3.

Procedure ***mapping*****.** The inputs of this procedure are g_{tem}, the genes in template pathway with available structural information, and g_{tar}, the related homologs in the target genome with the structural information among them. It will assign an ortholog for *each* gene in g_{tem} and output the ortholog pairs in a set *op*. Details are following.

1. Initialize g_{tem} as all the genes in template pathway with structural information, g_{tar} as the related homologs in target genome with structural information, *op* as empty. Repeat step (2) and (3) until g_{tem} is empty.
2. Call *OptimalOrth* on g_{tem} and g_{tar}, get an orthology assignment *orth*, which is a set of gene pairs (i, j) where $i \in g_{tem}$ and $j \in g_{tar}$. Put all the ortholog pairs in *orth* into *op*.
3. Remove all the template genes appeared in *orth* from g_{tem}, and their homologs from g_{tar}. If there is a *tf* gene whose product regulates some genes in the reduced g_{tem} removed, add it back to the reduce g_{tem} and its ortholog to the reduced g_{tar}. This is because that we still want the regulation information to constrain the orthology assignment for the reduced pathway.

Apparently, the procedure will output an orthology mapping such that the overall sequence similarity and the structural consistency achieves maximum.

2.3 Tree Decomposition Based Algorithm

Based on section 2.2, the orthology mapping with protein-DNA interactions and operon structures can be reduced to the problems of maximum independent set (MIS) on graphs formulated from the structural information. The problem is in general computationally intractable; any naive optimization algorithm would be very inefficient considering the pathway prediction is at the genome scale.

Our algorithm techniques are based on graph tree decomposition. A tree decomposition [15] of a graph provides a topological view on the graph and the tree width measures how much the graph is tree-like. Informally, in a tree decomposition, vertices from the original graph are grouped into a number of possibly intersecting bags; the bags topologically form a tree relationship. Shared vertices among intersecting bags form graph separators; efficient dynamic programming traversal over the graph is possible when all the bags are (*i.e.*, the tree width is) of small size [3].

In general, we solve the MIS problem on the graphs formulated from protein-DNA interactions and operon structures via tree decomposition based method (Procedure *mis*). On graphs that have small tree width, we employ the standard tree decomposition-based dynamic programming algorithm [3]. On graphs with larger tree width, we first find a tree decomposition on its complement graph, then find the maximum clique for the induced subgraph of each tree bag. Then the maximum independent set of the original graph must be the maximum of these maximum cliques. This is based two on facts: one is that a clique must be completely contained in a tree bag [4], thus the maximum of these maximum cliques for the subgraphs of the above mentioned complement graph must be the maximum clique of the complement graph; the other is that the maximum independent set of one graph is equivalent to the maximum clique of its complement graph. This way, the MIS problem could be transfered to the CLIQUE problem on smaller graphs. The CLIQUE problem is also solved via tree decomposition based method (Procedure *clique*).

In Procedure *mis*, we say the tree width is small if it is less than a predefined threshold *tw*. Procedure *clique* is recursive. The parameter *depth* denotes this is the $depth^{th}$ call (the initial call is the 1^{st} call). We define a density measure of a graph $G = (V, E)$ as $D = \frac{|E|}{\frac{(|V|-1)*|V|}{2}}$. If the density of the graph G is greater than a threshold d (e.g. 0.8), we say it is very dense. The procedures are described below.

Procedure *mis*(graph $G = (V, E)$):

1. Get tree width and tree decomposition of graph G.
2. If the tree width is small, find the maximum independent set of graph G by standard tree decomposition based method [3]. Return the maximum independent set.
3. If the tree width is not small, get the complement graph G_c of G, find a tree decomposition of G_c, find the maximum clique C_i for the induced subgraph G_{B_i} of each tree bag B_i by calling *clique*(1, G_{B_i}), return the maximum of all the C_i's.

Procedure *clique*(int *depth*, graph $G = (V, E)$):

1. If $depth \geq 3$ or G is very dense, find the maximum independent set of the complement of graph G by standard tree decomposition based method and return it.
2. Otherwise find a tree decomposition of G, find the maximum clique C_i for the induced subgraph G_{B_i} of each tree bag B_i by calling *clique*($depth + 1$, G_{B_i}), return the maximum of all the C_i's.

We use the parameter *depth* to restrict the recursive call of *clique* to a small number. According to our experiments, a threshold 3 can successfully reduce the tree width of the input graphs for the the standard tree decomposition-based dynamic programming algorithm used to find the maximum independent set. The running time of the standard tree decomposition-based algorithms is $O(2^t n)$, where t and n are respectively the tree width and the number of vertices in the graph. Since some biological networks (e.g. the E. coli network of regulatory interactions) follows a power-law distribution (i.e., few nodes are highly connected, whereas many have a low connectivity) [5], we expect our method is scalable to larger pathways and even biological networks. Due to the space limitation, we omit the formal definition of tree decomposition and the standard dynamic programming algorithm. Instead, we refer the reader to [3] for details.

We need to point out that finding the optimal tree decomposition (*i.e.*, the one with the smallest tree width) is NP-hard [2]. Since input graphs for Procedure *mis* and *clique* may not have small tree width, we use a simple, fast approximation algorithm greedy fill-in [6] to produce a tree decomposition and the approximated tree width for the given graphs. The approximated tree width t may affect the running time of the pathway prediction but not its accuracy. We reduce the tree width of the tree decompositions for dynamic programing by doing the transformation between MIS and CLIQUE described above.

3 Evaluation Results

We evaluated TdP against BH, BBH and PMAP by using 30 known pathways in *B. subtilis* 168 from KEGG pathway database [7] as templates (Table 1) to predict corresponding pathways in *E. coli* K12. For TdP, the operon structures are predicted according to the method used in [12] and experimentally confirmed transcriptional regulation information is taken from [8] for *B. subtilis* 168 and from [13] for *E. coli* K12. For PMAP, predicted operon and regulon information is obtained according to the method used in [9]. Both of TdP and PMAP include the COG filtering.

Table 1. Template pathways of *B. subtilis* 168, taken from KEGG pathway database

bsu00020	bsu00040	bsu00100	bsu00193	bsu00240	bsu00260	bsu00271	bsu00300
bsu00362	bsu00480	bsu00511	bsu00520	bsu00523	bsu00530	bsu00604	bsu00670
bsu00720	bsu00730	bsu00750	bsu00900	bsu00920	bsu00930	bsu01032	bsu02010
bsu02030	bsu02040	bsu03010	bsu03020	bsu03030	bsu03060		

We evaluated the accuracy of the algorithms. The accuracy was measured as the arithmetic mean of sensitivity and specificity. Let K be the real target pathway, H be the homologous genes searched by BLAST according to the corresponding template pathway. Let R be the size of $K \cap H$, *i.e.* the number of genes common in both the real target pathway and the candidate orthologues. We use this number as the number of real genes to calculate sensitivity and specificity because that is the maximum number of genes a sequence based method can predict correctly. Let TP (true positive) be the number of correctly predicted genes and FP (false positive) be the number of predicted genes that do not exist real target pathway. We define SE (sensitivity) as TP/RP, and SP (specificity) as $TP/(TP+FP)$. Since BH (or BBH) can be considered a subroutine of PMAP and TdP, we only evaluated efficiency for PMAP and TdP. Running times from reading inputs to output the predicted pathway were collected. For TdP, we also collected the data on tree width of the tree decompositions on the constructed graphs for dynamic programing. For all of the algorithms, program NCBI *blastp* [1] was used for BLAST search and the E-value threshold was set to 10^{-6}. The

Table 2. Evaluation results. SE: sensitivity, SP: specificity, A: accuracy ((SE+SP)/2), T: time (seconds).

	BH			BBH			TdP				PMAP			
	SE	SP	A	SE	SP	A	SE	SP	A	T	SE	SP	A	T
ave	0.807	0.892	0.849	0.779	0.926	0.853	0.82	0.928	0.874	14	0.849	0.878	0.864	16.5
min	0.333	0.333	0.333	0.333	0.5	0.45	0.39	0.428	0.514	0.8	0.5	0.5	0.633	12.8
max	1	1	1	1	1	1	1	1	1	159.9	1	1	1	52.8

experiments ran on a PC with 2.8 GHz Intel(R) Pentium 4 processor and 1-GB RAM, running RedHat Enterprise Linux version 4 AS. Running times were measured using the ”time” function. The testing results are summarized in Table 2.

On average, TdP has accuracy of 0.874, which is better than those of other algorithms. PMAP has the second highest accuracy, which means prediction accuracy could be improved even by incorporating structural information partially. We also give two examples here to show the improvement is good for small as well as large pathways. One is the aminosugars metabolism, which has 17 genes in *B. subtilis* 168 while 20 genes in *E. coli* K12. The prediction accuracy of TdP is 0.881, better than 0.839, 0.839 and 0.769 of BH, BBH and PMAP respectively. Another is the glycine, serine and threonine metabolism pathway, which has 36 genes both in *B. subtilis* 168 and *E. coli* K12. TdP has prediction accuracy of 0.858, better than 0.819, 0.779, 0.633 of BH, BBH and PMAP respectively.

For efficiency, TdP has average of 14.0 seconds for predicting a pathway, which is slightly better than 16.5 seconds of PMAP. The tree width distribution is shown in Figure 2. On average, tree width of the tree decompositions for dynamic programing is 3.7 and 93% of them have tree width at most 5. Since theoretically the running time to find the maximum independent set for an input graph by the tree decomposition based method is $O(2^t n)$ (where t is the tree width), we can conclude that most of the time our algorithm is efficient based on the statistics of the tree width.

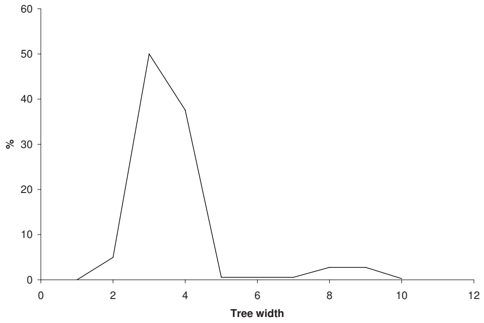


Fig. 2. Distribution of the tree width of the tree decompositions on the constructed graphs or their complement graphs

4 Discussion and Conclusion

We have shown our work in utilizing structural information including protein-DNA interactions and operon structures in comparative analysis based pathway prediction. The structural information used to constrain the orthology assignment between the template pathway and the one to be predicted appears to be critical for prediction accuracy improvement. It was to seek the sequence similarity and the structural consistency between the template and the predicted pathways as high as possible. Technically, the problem was formulated as finding the maximum independent set problem on the graphs constructed based on the structure information. Our algorithm, based on the non-trivial tree decomposition, coped with the computational intractability issue well and ran very efficiently. Evaluations on real pathway prediction for *E. coli* also showed the effectiveness of this approach. It could also utilize incomplete data and tolerate some noise in the data.

Tree decomposition based algorithm is sophisticated yet practically efficient. Simpler algorithms are possible if only functional information and sequence similarity are considered. However, computationally incorporating structure information such as protein-DNA interactions and operons in optimal pathway prediction appears to be inherently difficult. Naive optimization algorithms may not be scalable to larger pathway at the genome scale. In addition to the computational efficiency, our graph-theoretic approach also makes it possible to incorporate more information such as gene fusion and protein-protein interactions [14] to further improve the accuracy simply because such information may be represented as graphs as well.

On the other hand, when a template pathway is not well conserved in the target genome, the method may predict the pathway incorrectly. Multiple templates could be used to rescue this problem since the conserved information could be compensated with each other. We are trying to build profiles from multiple template pathways and use them to do the pathway prediction.

References

1. S. F. Altschul, T. L. Madden, A. A. Schffer, J. Zhang, Z. Zhang, W. Miller, D. J. Lipman, “Gapped BLAST and PSI-BLAST: a new generation of protein database search programs”, *Nucleic Acids Res*, 25, 3389-3402, 1997.
2. H. L. Bodlaender, “Classes of graphs with bounded tree-width”, *Tech. Rep. RUU-CS-86-22, Dept. of Computer Science, Utrecht University, the Netherlands*, 1986.
3. H. L. Bodlaender, “Dynamic programming algorithms on graphs with bounded tree-width”, In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, *Lecture Notes in Computer Science*, 317, 105-119, Springer Verlag, 1987.
4. H. L. Bodlaender. Discovering Treewidth. *SOFSEM*, 1-16, 2005.
5. R. M. Gutierrez-Rios, D. A. Rosenblueth, J. A. Loza, A. M. Huerta, J. D. Glasner, F. R. Blattner, J. Collado-Vides, “Regulatory network of Escherichia coli: consistency between literature knowledge and microarray profiles”, Genome Res. 13(11), 2435-2443, 2003.

6. I. V. Hicks, A. M. C. A. Koster, E. Kolotoglu, "Branch and tree decomposition techniques for discrete optimization", In *Tutorials in Operations Research: INFORMS – New Orleans*, 2005.
7. M. Kanehisa, S. Goto, M. Hattori, K. F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, M. Hirakawa, "From genomics to chemical genomics: new developments in KEGG", *Nucleic Acids Res.* 34, D354-357, 2006.
8. Y. Makita, M. Nakao, N. Ogasawara, K. Nakai, "DBTBS: database of transcriptional regulation in Bacillus subtilis and its contribution to comparative genomics", *Nucleic Acids Res.*, 32, D75-77, 2004
9. F. Mao, Z. Su, V. Olman, P. Dam, Z. Liu, Y. Xu, "Mapping of orthologous genes in the context of biological pathways: An application of integer programming", *PNAS*, 108 (1), 129-134, 2006.
10. D. W. Mount, *Bioinformatics: sequence and genome analysis*, Cold Spring Harbor Lab Press, 516-517, 2000.
11. R. Nielsen, "Comparative genomics: Difference of expression", *Nature*, 440, 161-161, 2006.
12. M. N. Price, K. H. Huang, E. J. Alm, A. P. Arkin, "A novel method for accurate operon predictions in all sequenced prokaryotes", *Nucleic Acids Res.*, 33, 880-892, 2005.
13. H. Salgado, S. Gama-Castro, M. Peralta-Gil, etc., "RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions", *Nucleic Acids Res.*, 34, D394-D397, 2006.
14. J. L. Reed, I. Famili, I. Thiele, B. O. Palsson, "Towards multidimensional genome annotation.", *Nature Reviews Genetics*, 7, 130-141, 2006.
15. N. Robertson and P. D. Seymour, "Graph minors ii. algorithmic aspects of tree width", *J. Algorithms*, 7, 309-322, 1986.
16. P. Romero, J. Wagg, M. L. Green, D. Kaiser, M. Krummenacker, P. D. Karp, "Computational prediction of human metabolic pathways from the complete human genome", *Genome Biology*, 6, R2, 2004.
17. Z. Su, P. Dam, X. Chen, V. Olman, T. Jiang, B. Palenik, Y. Xu, "Computational Inference of Regulatory Pathways in Microbes: an Application to Phosphorus Assimilation Pathways in Synechococcus sp. WH8102", *Genome Informatics*, 14, 3-13, 2003.
18. R. L. Tatusov, E. V. Koonin, D. J. Lipman, "A Genomic Perspective on Protein Families", *Science*, 278 (5338), 631-637, 1997.

Extending the Calculus of Looping Sequences to Model Protein Interaction at the Domain Level

Roberto Barbuti, Andrea Maggiolo–Schettini, and Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 - Pisa, Italy
{barbuti,maggiolo,milazzo}@di.unipi.it

Abstract. In previous papers we introduced a formalism, called Calculus of Looping Sequences (CLS), for describing biological systems and their evolution. CLS is based on term rewriting. Terms can be constructed by composing symbols of a given alphabet in sequences, which could be closed (looping) and contain other terms. In this paper we extend CLS to represent protein interaction at the domain level. Such an extension, called Calculus of Linked Looping Sequences (LCLS), is obtained by labeling alphabet symbols used in terms. Two symbols with the same label are considered to be linked. We introduce a type system to express a concept of well–formedness of LCLS terms, we give an operational semantics of the new calculus, and we show the application of LCLS to the description of a biological system.

1 Introduction

Among the formalisms that either have been applied to or have been inspired by biological systems there are automata–based models [1,8], rewrite systems [5,9], and process calculi [11,10,4]. Automata have the advantage of allowing the direct use of many verification tools such as model checkers. Rewrite systems usually allow describing biological systems with a notation that can be easily understood by biologists. On the other hand, automata–like models and rewrite systems present, in general, problems of compositionality. Compositionality allows studying the behavior of a system componentwise, and it is in general ensured by process calculi, included those used to describe biological systems.

In [2,3] we introduced a new formalism, called Calculus of Looping Sequences (CLS for short), for describing biological systems and their evolution. CLS is based on term rewriting with some features, such as a commutative parallel composition operator, and some semantic means, such as bisimulations, that are common in process calculi. This permits to combine the simplicity of notation of rewriting systems with the advantage of a form of compositionality. Actually, in [3] we have defined bisimulation relations which are congruences with respect to the operators. This is ensured by the assumption that the same set of rewrite rules is used for terms that are composed.

CLS terms are constructed by starting from basic constituent elements and composing them by means of operators of sequencing, looping, containment and

I. Măndoiu and A. Zelikovsky (Eds.): ISBRA 2007, LNBI 4463, pp. 638–649, 2007.

parallel composition. Sequences may represent DNA fragments and proteins, looping sequences may represent membranes, and parallel composition may represent juxtaposition.

A formalism for modelling protein interactions was developed in the seminal paper by Danos and Laneve [5], and extended in [6]. This formalism allows expressing proteins by a node with a fixed number of domains; binding between domains allows complexating proteins. In this work we extend CLS to represent protein interaction at the domain level. Such an extension, called Calculus of Linked Looping Sequences (LCLS), is obtained by labelling elements of sequences. Two elements with the same label are considered to be linked.

We introduce a type system to express a concept of well–formedness of LCLS terms, we give an operational semantics of the new calculus, and, finally, we show the application of LCLS to the description of a biological system.

2 The Calculus of Looping Sequences

In this section we recall the Calculus of Looping Sequences (CLS). It is essentially based on term rewriting, hence a CLS model consists of a term and a set of rewrite rules. The term is intended to represent the structure of the modeled system, and the rewrite rules the events that may cause the system to evolve.

We start with defining the syntax of terms. We assume a possibly infinite alphabet $\mathcal{E}$ of symbols ranged over by $a, b, c, \ldots$.

Definition 1 (Terms). Terms T *and* Sequences S *of CLS are given by the following grammar:*

$$\begin{array}{rcl} T & ::= & S \quad \big| \quad \left(S\right)^{L} \rfloor\, T \quad \big| \quad T \,|\, T \\ S & ::= & \epsilon \quad \big| \quad a \quad \big| \quad S \cdot S \end{array}$$

where a is a generic element of $\mathcal{E}$, and ϵ represents the empty sequence. We denote with $\mathcal{T}$ the infinite set of terms, and with $\mathcal{S}$ the infinite set of sequences.

In CLS we have a sequencing operator $\_ \cdot \_$, a looping operator $\left(\_\right)^{L}$, a parallel composition operator $\_ \,|\, \_$ and a containment operator $\_ \rfloor \_$. Sequencing can be used to concatenate elements of the alphabet $\mathcal{E}$. The empty sequence ϵ denotes the concatenation of zero symbols. A term can be either a sequence, or a looping sequence (that is the application of the looping operator to a sequence) containing another term, or the parallel composition of two terms. By definition, looping and containment are always applied together, hence we can consider them as a single binary operator $\left(\_\right)^{L} \rfloor \_$ which applies to one sequence and one term.

The biological interpretation of the operators is the following: the main entities which occurs in cells are DNA and RNA strands, proteins, membranes, and other macro–molecules. DNA strands (and similarly RNA strands) are sequences of nucleic acids, but they can be seen also at a higher level of abstraction as sequences of genes. Proteins are sequence of amino acids which usually have a very complex three–dimensional structure. In a protein there are usually (relatively) few subsequences, called domains, which actually are able to interact with other

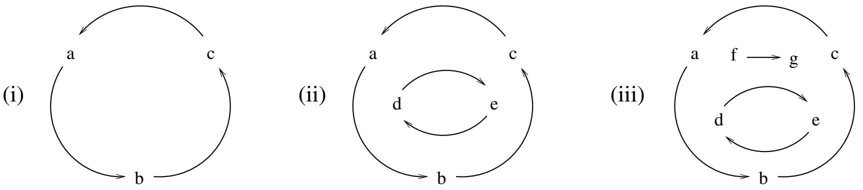


Fig. 1. (i) represents $\big(a \cdot b \cdot c\big)^L$; (ii) represents $\big(a \cdot b \cdot c\big)^L \rfloor \big(d \cdot e\big)^L$; (iii) represents $\big(a \cdot b \cdot c\big)^L \rfloor (\big(d \cdot e\big)^L \mid f \cdot g)$

entities by means of chemical reactions. CLS sequences can model DNA/RNA strands and proteins by describing each gene or each domain with a symbol of the alphabet. Membranes are closed surfaces, often interspersed with proteins, which may contain something. A closed surface can be modeled by a looping sequence. The elements (or the subsequences) of the looping sequence may represent the proteins on the membrane, and by the containment operator it is possible to specify the content of the membrane. Other macro–molecules can be modeled as single alphabet symbols, or as short sequences. Finally, juxtaposition of entities can be described by the parallel composition of their representations.

Brackets can be used to indicate the order of application of the operators, and we assume $\big(\_\big)^L \rfloor \_$ to have precedence over $\_ \mid \_$. In Figure 1 we show some examples of CLS terms and their visual representation.

In CLS we may have syntactically different terms representing the same structure. We now introduce a structural congruence relation to identify such terms.

Definition 2 (Structural Congruence). *The structural congruence relations $\equiv_S$ and $\equiv_T$ are the least congruence relations on sequences and on terms, respectively, satisfying the following rules:*

$$S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S$$

$$S_1 \equiv_S S_2 \textit{ implies } S_1 \equiv_T S_2 \textit{ and } \big(S_1\big)^L \rfloor T \equiv_T \big(S_2\big)^L \rfloor T$$

$$T_1 \mid T_2 \equiv_T T_2 \mid T_1 \qquad T_1 \mid (T_2 \mid T_3) \equiv_T (T_1 \mid T_2) \mid T_3 \qquad T \mid \epsilon \equiv_T T$$

$$\big(\epsilon\big)^L \rfloor \epsilon \equiv_T \epsilon \qquad \big(S_1 \cdot S_2\big)^L \rfloor T \equiv_T \big(S_2 \cdot S_1\big)^L \rfloor T$$

Rules of the structural congruence state the associativity of $\cdot$ and $\mid$, the commutativity of the latter and the neutral role of ϵ. Moreover, axiom $\big(S_1 \cdot S_2\big)^L \rfloor T \equiv_T \big(S_2 \cdot S_1\big)^L \rfloor T$ says that looping sequences can rotate. In the following, for simplicity, we will use $\equiv$ in place of $\equiv_T$.

Rewrite rules will be defined essentially as pairs of terms, in which the first term describes the portion of the system in which the event modeled by the rule may occur, and the second term describes how that portion of the system changes when the event occurs. In the terms of a rewrite rule we allow the use of variables. As a consequence, a rule will be applicable to all terms which can be obtained by properly instantiating its variables. Variables can be of three kinds: two of these are associated with the two different syntactic categories of terms and sequences, and one is associated with single alphabet elements. We assume a set of term variables TV ranged over by $X, Y, Z, \ldots$, a set of sequence variables SV ranged over by $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$, and a set of element variables $\mathcal{X}$ ranged over by $x, y, z, \ldots$. All these sets are possibly infinite and pairwise disjoint. We denote by $\mathcal{V}$ the set of all variables, $\mathcal{V} = TV \cup SV \cup \mathcal{X}$, and with ρ a generic variable of $\mathcal{V}$. Hence, a pattern is a term which may include variables.

Definition 3 (Patterns). Patterns P *and* sequence patterns SP *of CLS are given by the following grammar:*

$$\begin{array}{rcl} P & ::= & SP \quad \big| \quad \left(SP\right)^{L} \rfloor\, P \quad \big| \quad P \mid P \quad \big| \quad X \\ SP & ::= & \epsilon \quad \big| \quad a \quad \big| \quad SP \cdot SP \quad \big| \quad \widetilde{x} \quad \big| \quad x \end{array}$$

where a is a generic element of $\mathcal{E}$, and $X, \widetilde{x}$ and x are generic elements of TV, SV and $\mathcal{X}$, respectively. We denote with $\mathcal{P}$ the infinite set of patterns.

We assume the structural congruence relation to be trivially extended to patterns. An *instantiation* is a partial function $\sigma : \mathcal{V} \to \mathcal{T}$. An instantiation must preserve the type of variables, thus for $X \in TV, \widetilde{x} \in SV$ and $x \in \mathcal{X}$ we have $\sigma(X) \in \mathcal{T}, \sigma(\widetilde{x}) \in \mathcal{S}$ and $\sigma(x) \in \mathcal{E}$, respectively. Given $P \in \mathcal{P}$, with $P\sigma$ we denote the term obtained by replacing each occurrence of each variable $\rho \in \mathcal{V}$ appearing in P with the corresponding term $\sigma(\rho)$. With Σ we denote the set of all the possible instantiations and, given $P \in \mathcal{P}$, with $Var(P)$ we denote the set of variables appearing in P. Now we define rewrite rules.

Definition 4 (Rewrite Rules). *A rewrite rule is a pair of patterns (P_1, P_2), denoted with $P_1 \mapsto P_2$, where $P_1, P_2 \in \mathcal{P}$, $P_1 \not\equiv \epsilon$ and such that $Var(P_2) \subseteq Var(P_1)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.*

A rewrite rule $P_1 \mapsto P_2$ states that a term $P_1\sigma$, obtained by instantiating variables in P_1 by some instantiation function σ, can be transformed into the term $P_2\sigma$. We define the semantics of CLS as a transition system, in which states correspond to terms, and transitions correspond to rule applications.

Definition 5 (Semantics). *Given a set of rewrite rules $\mathcal{R} \subseteq \Re$, the* semantics *of CLS is the least transition relation $\to$ on terms closed under $\equiv$, and satisfying the following inference rules:*

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma}{P_1\sigma \to P_2\sigma} \qquad \frac{T_1 \to T_2}{T \mid T_1 \to T \mid T_2} \qquad \frac{T_1 \to T_2}{\left(S\right)^{L} \rfloor\, T_1 \to \left(S\right)^{L} \rfloor\, T_2}$$

where the symmetric rule for the parallel composition is omitted.

A *model* in CLS is given by a term describing the initial state of the system and by a set of rewrite rules describing all the events that may occur.

3 The Calculus of Linked Looping Sequences

To model a protein at the domain level in CLS it would be natural to use a sequence with one symbol for each domain. However, the binding between two domains of two different proteins, that is the linking between two elements of two different sequences, cannot be expressed in CLS. To represent this, we extend CLS by labels on basic symbols. If in a term two symbols have the same label, we intend that they represent domains that are bound to each other. If in a term there is a single symbol with a certain label, we intend that the term represents only a part of a system we model, and that the symbol will be linked to another symbol in another part of the term representing the full model.

As membranes create compartments, elements inside a looping sequence cannot be linked to elements outside. Elements inside a membrane can be linked either to other elements inside the membrane or to elements of the membrane itself. An element can be linked at most to another element. The partner to which an element is bound can be different at different times, and a domain able to bind to multiple partners simultaneously could be described by using more elements instead of a single one.

The syntax of terms of the Calculus of Linked Looping Sequences (LCLS) is defined as follows. We use as labels natural numbers.

Definition 6 (Terms). Terms T *and* Sequences S *of LCLS are given by the following grammar:*

$$\begin{array}{rcl} T & ::= & S \quad \big| \quad \left(S\right)^{L} \rfloor T \quad \big| \quad T \mid T \\ S & ::= & \epsilon \quad \big| \quad a \quad \big| \quad a^{n} \quad \big| \quad S \cdot S \end{array}$$

where a is a generic element of $\mathcal{E}$, and n is a natural number. We denote with $\mathcal{T}$ the infinite set of terms, and with $\mathcal{S}$ the infinite set of sequences.

The structural congruence relation is the same as for CLS. Patterns of LCLS are similar to those of CLS, with the addition of the labels.

Definition 7 (Patterns). Patterns P *and* sequence patterns SP *of LCLS are given by the following grammar:*

$$\begin{array}{rcl} P & ::= & SP \quad \big| \quad \left(SP\right)^{L} \rfloor P \quad \big| \quad P \mid P \quad \big| \quad X \\ SP & ::= & \epsilon \quad \big| \quad a \quad \big| \quad a^{n} \quad \big| \quad SP \cdot SP \quad \big| \quad \widetilde{x} \quad \big| \quad x \quad \big| \quad x^{n} \end{array}$$

where a is an element of $\mathcal{E}$, n is a natural number and $X, \widetilde{x}$ and x are elements of TV, SV and $\mathcal{X}$, respectively. We denote with $\mathcal{P}$ the infinite set of patterns.

Note that an LCLS term is also an LCLS pattern; everything we define for patterns will be immediately defined also for terms. Moreover, in what follows, we will often use the notions of *compartment* and of *top–level compartment* of a pattern. A compartment is a subpattern that is the content of a looping sequence and in which the contents of inner looping sequences are not considered. The top–level compartment is the portion of the pattern that is not inside any looping sequence. For instance, the top–level compartment of a pattern $P = a \mid \left(b\right)^{L} \rfloor c \mid \left(d\right)^{L} \rfloor (X \mid \left(e\right)^{L} \rfloor f)$ is $a \mid \left(b\right)^{L} \rfloor \epsilon \mid \left(d\right)^{L} \rfloor \epsilon$. Other compartments in P are c, $X \mid \left(e\right)^{L} \rfloor \epsilon$, and f.

An LCLS pattern is well–formed if and only if a label occurs no more than twice, and two occurrences of a label are always in the same compartment. The following type system will be used for deriving the well–formedness of patterns.

In each inference rule the conclusion has the form $(N, N') \models P$, where N and N' are sets of natural numbers with N the set of labels used twice and N' the set of labels used only once in the top–level compartment of P.

Definition 8 (Type System). *The typing algorithm for LCLS patterns is defined by the following inference rules:*

$$1.\ (\varnothing,\varnothing) \models \epsilon \qquad 2.\ (\varnothing,\varnothing) \models a \qquad 3.\ (\varnothing,\{n\}) \models a^n$$

$$4.\ (\varnothing,\varnothing) \models x \qquad 5.\ (\varnothing,\{n\}) \models x^n \qquad 6.\ (\varnothing,\varnothing) \models \widetilde{x} \qquad 7.\ (\varnothing,\varnothing) \models X$$

$$8.\ \frac{(N_1,N_1') \models SP_1 \quad (N_2,N_2') \models SP_2 \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing}{(N_1 \cup N_2 \cup (N_1' \cap N_2'), (N_1' \cup N_2') \setminus (N_1' \cap N_2')) \models SP_1 \cdot SP_2}$$

$$9.\ \frac{(N_1,N_1') \models P_1 \quad (N_2,N_2') \models P_2 \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing}{(N_1 \cup N_2 \cup (N_1' \cap N_2'), (N_1' \cup N_2') \setminus (N_1' \cap N_2')) \models P_1 \mid P_2}$$

$$10.\ \frac{(N_1,N_1') \models SP \quad (N_2,N_2') \models P \quad N_1 \cap N_2 = N_1' \cap N_2 = N_1 \cap N_2' = \varnothing \quad N_2' \subseteq N_1'}{(N_1 \cup N_2', N_1' \setminus N_2') \models \left(SP\right)^L \rfloor P}$$

where a is a generic element of $\mathcal{E}$, n is a natural number, and $X, \widetilde{x}$ and x are generic elements of TV, SV and $\mathcal{X}$, respectively. We write $\models P$ if there exist $N, N' \subset \mathbb{N}$ such that $(N, N') \models P$, and $\not\models P$ otherwise.

Rules 1–7 are self explanatory. Rule 8 states that a sequence pattern $SP_1 \cdot SP_2$ is well–typed if there are no variables occurring either four times ($N_1 \cap N_2 = \varnothing$) or three times ($N_1' \cap N_2 = N_1 \cap N_2' = \varnothing$). Variables occurring twice in $SP_1 \cdot SP_2$ are those which occur twice either in SP_1 or in SP_2 together with variables occurring once both in SP_1 and in SP_2. Rule 9 for the parallel composition is analogous to rule 8. Rule 10 states that the only labels which can be used for typing $\left(SP\right)^L \rfloor P$ must be different from those used for typing P. Moreover the labels used once in P must be used once in SP, that is these labels are used to bind elements inside the membrane to elements on the membrane itself.

The following lemma states some simple properties of the type system.

Lemma 1. *Given $N, N' \subset \mathbb{N}$, and $P \in \mathcal{P}$, then $(N, N') \models P$ implies:*
(i) *both N and N' are finite;* (ii) *$N \cap N' = \varnothing$.*

Definition 9 (Well–Formedness of Patterns). *A pattern P is* well–formed *if and only if $\models P$ holds.*

Now we give two lemmas. The first relates the well–formedness of a pattern with the well–formedness of its subpatterns. The second states that well–formedness is preserved by structural congruence.

Lemma 2. *Given $P \in \mathcal{P}$ and P' a subpattern of P, then $\models P$ implies $\models P'$.*

Lemma 3. *Given $P_1, P_2 \in \mathcal{P}$, $\models P_1$ and $P_1 \equiv P_2$ imply $\models P_2$.*

The use of labels to represent links is not new. In [5] well–formedness of terms is given by a concept of graph–likeness. We notice that in our case membranes, which are not present in the formalism of [5], make the treatment more complicated. In [6], where the concept of membrane is introduced, well–formedness of terms is given intuitively and not formally defined.

We say that a well–formed pattern P is *closed* if and only if $(N, \varnothing) \models P$ for some $N \subset \mathbb{N}$, and that it is *open* otherwise. Moreover, we say that P is *link–free* if and only if $(\varnothing, \varnothing) \models P$. Since patterns include terms, we use the same terminology also for terms. For example, $a \cdot b \cdot c \mid d \cdot x$ is a link–free pattern, $a \cdot b^1 \cdot c \mid d \cdot x^1$ is a closed pattern, and $a \cdot b^1 \cdot c^2 \mid d \cdot x^1$ is an open pattern.

In the following we shall use a notion of set of links of a pattern, namely the set of labels that occur twice in the top–level compartment of the pattern.

Definition 10. *The* set of links *of a pattern P is $L(P) = \{n | \#(n, L_M(P)) = 2\}$, where $L_M(P)$ is the* multiset of labels *of P, recursively defined as follows:*

$$L_M(\epsilon) = \varnothing \qquad L_M(\nu) = \varnothing \qquad L_M(\nu^n) = \{n\} \qquad L_M(\widetilde{x}) = \varnothing$$

$$L_M(SP_1 \cdot SP_2) = L_M(SP_1) \cup L_M(SP_2) \qquad L_M(P_1 \mid P_2) = L_M(P_1) \cup L_M(P_2)$$

$$L_M(\left(SP\right)^L \rfloor P) = L_M(SP) \cup (L_M(SP) \cap L_M(P)) \qquad L_M(X) = \varnothing$$

where $\nu \in \mathcal{E} \cup EV$, $n \in \mathbb{N}$, P_1, P_2 are any pattern, SP is any sequence pattern.

If P is a well–formed pattern, there exists $N \subset \mathbb{N}$ such that $(L(P), N) \models P$.

Let $\mathcal{A}$ be the set of all total injective functions $\alpha : \mathbb{N} \to \mathbb{N}$. Given $\alpha \in \mathcal{A}$, the *α–renaming* of an LCLS pattern P is the pattern $P\alpha$ obtained by replacing every label n in P by $\alpha(n)$. It holds that α–renaming preserves well–formedness.

Lemma 4. *Given $P \in \mathcal{P}$, $\forall \alpha \in \mathcal{A}$ it holds $\models P \iff \models P\alpha$.*

Links in a term are placeholders: the natural number used in the two labels of a link has not a particular meaning. Hence, we can consider as equivalent patterns which differ only in the values of their links.

Definition 11 (α–equivalence). *The α–equivalence relation $=_\alpha$ on LCLS patterns is the least equivalence relation which satisfies the following rules:*

$$\nu^{n1} \mid \mu^{n1} =_\alpha \nu^{n2} \mid \mu^{n2} \qquad \frac{P_1 \mid P_2 =_\alpha P_3}{P_2 \mid P_1 =_\alpha P_3} \qquad \frac{SP_1 \mid SP_2 =_\alpha P_3}{SP_1 \cdot SP_2 =_\alpha P_3}$$

$$\frac{P_1 =_\alpha P_2 \quad P_3 =_\alpha P_4 \quad L(P_1) \cap L(P_3) = L(P_2) \cap L(P_4) = \varnothing}{P_1 \mid P_3 =_\alpha P_2 \mid P_4}$$

$$\frac{SP_1 =_\alpha SP_2 \quad P_1 =_\alpha P_2 \quad L(SP_1) \cap L(SP_2) = L(P_1) \cap L(P_2) = \varnothing}{\left(SP_1\right)^L \rfloor P_1 =_\alpha \left(SP_2\right)^L \rfloor P_2}$$

$$\frac{\left(SP_1 \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \left(SP_2 \cdot SP_2'\right)^L \rfloor P_2 \quad ni \notin L_M(SP_i \cdot SP_i') \cup L_M(P_i)}{\left(SP_1 \cdot \nu^{n1} \cdot SP_1'\right)^L \rfloor (\mu^{n1} \mid P_1) =_\alpha \left(SP_2 \cdot \nu^{n2} \cdot SP_2'\right)^L \rfloor (\mu^{n2} \mid P_2)}$$

$$\frac{\left(SP_1 \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \left(SP_2 \cdot SP_2'\right)^L \rfloor P_2 \quad ni \notin L_M(SP_i \cdot SP_i') \cup L_M(P_i)}{\nu^{n1} \mid \left(SP_1 \cdot \mu^{n1} \cdot SP_1'\right)^L \rfloor P_1 =_\alpha \nu^{n2} \mid \left(SP_2 \cdot \mu^{n2} \cdot SP_2'\right)^L \rfloor P_2}$$

where $\nu, \mu \in \mathcal{E} \cup EV$, $n1, n2 \in \mathbb{N}$, P_1, P_2, P_3, P_4 are any pattern, SP_1, SP_2, SP_3, SP_4 are any sequence pattern.

It is easy to see that α–equivalence preserves well–formedness of patterns.

Lemma 5. *Given $P_1, P_2 \in \mathcal{P}$, $\models P_1$ and $P_1 =_\alpha P_2$ imply $\models P_2$.*

Note that the labels occurring only once in a pattern P are not renamed by the α–equivalence relation. Instead, the application of an α–renaming function to P may change these labels. Moreover, labels which occur twice in more than one compartment of the pattern can be renamed differently in each compartment by the α–equivalence relation, while they are all renamed by the same value by applying some α–renaming function.

We say that an instantiation function σ is well–formed if it maps variables into well–formed closed terms and sequences. We denote with Σ_{wf} the set of all well–formed instantiation functions. Differently from CLS, the application of an instantiation function to a pattern does not correspond to the substitution of every variable in the pattern with the corresponding term given by the instantiation function, because this could lead to not well–formed terms. As an example, consider the well-formed pattern $P = a \cdot \widetilde{x} \mid X$ and a well–formed instantiation function σ such that $\sigma(\widetilde{x}) = b^1 \cdot c^1$ and $\sigma(X) = d^1 \mid e^1$. The application of σ to P would produce the term $P\sigma = a \cdot b^1 \cdot c^1 \mid d^1 \mid e^1$, which is not well–formed. Similarly, consider the well–formed pattern $P = a \cdot \widetilde{x} \cdot \widetilde{x}$ and the same well-formed instantiation function. We obtain $P\sigma = a \cdot b^1 \cdot c^1 \cdot b^1 \cdot c^1$, which is not well–formed. To avoid these situations, we define the application of an instantiation function to an LCLS pattern in a way such that the links in the instantiations of all occurrences of all variables are renamed, if necessary.

Definition 12 (Pattern Instantiation). *Given a pattern $P \in \mathcal{P}$ and an instantiation function $\sigma \in \Sigma$, the application of σ to P is an LCLS term $P\sigma$ given by the following inductive definition:*

$$\epsilon\sigma = \epsilon \quad a\sigma = a \quad a^n\sigma = a^n \quad \widetilde{x}\sigma = \sigma(\widetilde{x}) \quad x\sigma = \sigma(x) \quad x^n\sigma = \sigma(x)^n \quad X\sigma = \sigma(X)$$

$$\frac{SP_i\sigma =_\alpha S_i \quad L(S_1) \cap L(S_2) = \varnothing}{SP_1 \cdot SP_2\ \sigma = S_1 \cdot S_2} \qquad \frac{P_i\sigma =_\alpha T_i \quad L(T_1) \cap L(T_2) = \varnothing}{P_1 \mid P_2\ \sigma = T_1 \mid T_2}$$

$$\frac{SP\sigma =_\alpha S \quad P\sigma =_\alpha T \quad L(S) \cap L(T) = \varnothing}{\left(SP\right)^L \rfloor P\ \sigma = \left(S\right)^L \rfloor T}$$

where P_1, P_2, P are any pattern, SP_1, SP_2, SP are any sequence pattern.

Now, by applying a well–formed instantiation function to a well–formed pattern, we obtain a well–formed term.

Lemma 6. *Given $P \in \mathcal{P}, \sigma \in \Sigma_{wf}$, it holds that $\models P$ implies $\models P\sigma$.*

As in CLS, rewrite rules in LCLS are pairs of patterns.

Definition 13 (Rewrite Rules). *A rewrite rule is a pair of patterns (P_1, P_2), denoted with $P_1 \mapsto P_2$, where $P_1, P_2 \in \mathcal{P}$, $P_1 \not\equiv \epsilon$ and such that $Var(P_2) \subseteq Var(P_1)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.*

Our aim is to show that the application of a rewrite rule composed by well–formed patterns to a well–formed term produces another well–formed term. It is easy to see that, as a consequence of Lemma 6, this holds if variables of the rewrite rule are instantiated by a well–formed instantiation function. However, sometimes we would like to relax this constraint and allow a variable to be instantiated with an open term. For instance, we would permit the application of a rewrite rule $\widetilde{x} \cdot a \mapsto \widetilde{x} \cdot b$ to the term $c^1 \mid d^1 \cdot a$ (so to obtain $c^1 \mid d^1 \cdot b$), which requires that $\sigma(\widetilde{x}) = d^1$. Relaxing this constraint causes the introduction of constraints on the two patterns of the rewrite rules: they must not add or remove occurrences of variables, they cannot move variables from a compartment to another one, and they cannot add single occurrences of labels. To check these constraints we introduce a notion of compartment safety.

Definition 14 (Compartment Safety). *The* compartment safety relation *cs on pairs of patterns is the least equivalence relation satisfying the following rules:*

$$cs(\epsilon,\epsilon) \qquad cs(\epsilon,\nu) \qquad cs(\nu^n,\mu^n) \qquad cs(\epsilon,\nu^n|\mu^n) \qquad cs(\widetilde{x},\widetilde{x}) \qquad cs(X,X)$$

$$\frac{cs(P_1,P_2) \quad cs(P_3,P_4)}{cs(P_1|P_3,P_2|P_4)} \qquad \frac{cs(P_1|P_2,P_3)}{cs(P_2|P_1,P_3)} \qquad \frac{cs(SP_1|SP_2,P_3)}{cs(SP_1 \cdot SP_2,P_3)}$$

$$\frac{cs(SP_1,SP_2) \quad cs(P_1,P_2)}{cs((SP_1)^L \rfloor P_1,(SP_2)^L \rfloor P_2)} \qquad \frac{cs((SP_1 \cdot SP_2)^L \rfloor P_1,(SP_3)^L \rfloor P_2)}{cs((SP_2 \cdot SP_1)^L \rfloor P_1,(SP_3)^L \rfloor P_2)}$$

$$\frac{cs((SP_1)^L \rfloor P_1,(SP_2)^L \rfloor P_2)}{cs((SP_1)^L \rfloor P_1,(SP_2 \cdot \nu^n)^L \rfloor (\mu^n \mid P_2))} \qquad \frac{cs((SP_1)^L \rfloor P_1,(SP_2)^L \rfloor P_2)}{cs((SP_1)^L \rfloor P_1,\nu^n \mid (SP_2 \cdot \mu^n)^L \rfloor P_2)}$$

where $\nu, \mu \in \mathcal{E} \cup EV$, $n \in \mathbb{N}$, P_1, P_2, P_3, P_4 are any pattern, SP_1, SP_2, SP_3 are any sequence pattern.

Definition 15 (Compartment Safe Rewrite Rule). *A rewrite rule $P_1 \mapsto P_2$ is* compartment safe (CS) *if $cs(P_1, P_2)$ holds. It is* compartment unsafe (CU) *otherwise. We denote with $\Re^{CS} \subset \Re$ the infinite set of CS rewrite rules, and with $\Re^{CU} \subset \Re$ the infinite set of CU rewrite rules.*

Now, we can introduce well–formedness also for rewrite rules.

Definition 16 (Well–Formedness of Rewrite Rules). *Given a rewrite rule $P_1 \mapsto P_2 \in \Re$, it is* well–formed *if P_1 and P_2 are well–formed patterns, and either $P_1 \mapsto P_2 \in \Re^{CS}$ or both P_1 and P_2 are closed patterns.*

The application of a well–formed rule satisfying compartment safety to a well–formed term preserves the well–formedness of the term even if variables are instantiated by a non well–formed instantiation function.

Lemma 7. *Given $\sigma \in \Sigma$ and a well–formed rewrite rule $P_1 \mapsto P_2$ such that $P_1 \mapsto P_2 \in \Re^{CS}$, it holds that $\models P_1\sigma$ implies $\models P_2\sigma$.*

Now, we can define the semantics of LCLS.

Definition 17 (Semantics). *Given a set of rewrite rules $\mathcal{R} \subseteq \Re$, such that $\mathcal{R} = \mathcal{R}^{CS} \cup \mathcal{R}^{CU}$ with $\mathcal{R}^{CS} \subset \Re^{CS}$ and $\mathcal{R}^{CU} \subset \Re^{CU}$, the* semantics *of LCLS is the least transition relation $\rightarrow$ on terms closed under $\equiv$ and $=_\alpha$, and satisfying the following inference rules:*

$$\textit{(appCS)}\quad \frac{P_1 \mapsto P_2 \in \mathcal{R}^{CS} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma \quad \alpha \in \mathcal{A}}{P_1\alpha\sigma \rightarrow P_2\alpha\sigma}$$

$$\textit{(appCU)}\quad \frac{P_1 \mapsto P_2 \in \mathcal{R}^{CU} \quad P_1\sigma \not\equiv \epsilon \quad \sigma \in \Sigma_{wf} \quad \alpha \in \mathcal{A}}{P_1\alpha\sigma \rightarrow P_2\alpha\sigma}$$

$$\textit{(par)}\quad \frac{T_1 \rightarrow T_1' \qquad L(T_1) \cap L(T_2) = \{n_1, \ldots, n_M\} \qquad n_1', \ldots, n_M' \textit{ fresh}}{T_1 \mid T_2 \rightarrow T_1'\{^{n_1'}, \ldots, ^{n_M'}/n_1, \ldots, n_M\} \mid T_2}$$

$$\textit{(cont)}\quad \frac{T \rightarrow T' \qquad L(S) \cap L(T') = \{n_1, \ldots, n_M\} \qquad n_1', \ldots, n_M' \textit{ fresh}}{(S)^L \rfloor T \rightarrow (S)^L \rfloor T'\{^{n_1'}, \ldots, ^{n_M'}/n_1, \ldots, n_M\}}$$

where the symmetric rule for the parallel composition is omitted.

Rules *(appCS)* and *(appCU)* describe the application of compartment safe and compartment unsafe rewrite rules, respectively. In the latter case we require that the instantiation function used to apply the rule is well–formed. In both cases, an α–renaming function is used to rename the labels in the pattern, in particular those appearing only once in the top–level compartment. The *(par)* and *(cont)* rules propagate the effect of a rewrite rule application to contexts by resolving conflicts in the use of labels.

Finally, we can give a theorem which states that the application of well–formed rewrite rules to well–formed terms produces new well–formed terms.

Theorem 1 (Subject Reduction). *Given a set of well–formed rewrite rules $\mathcal{R}$ and $T \in \mathcal{T}$, it holds that $\models T$ and $T \rightarrow T'$ imply $\models T'$.*

4 An Example: The EGF Signalling Pathway

A cell recognizes the EGF signal from the environment because it has on its membrane some EGF receptor proteins (EGFR), which are transmembrane proteins (they have some intra–cellular and some extra–cellular domains). One of the extra–cellular domains binds to one EGF protein in the environment, forming a signal–receptor complex on the membrane. This causes a conformational change on the receptor protein that enables it to bind to another one signal–receptor complex. The formation of the binding of the two signal–receptor complexes (called dimerization) causes the phosphorylation of some intra–cellular domains of the dimer. This causes the internal domains of the dimer to be recognized by a protein that is in the cytoplasm, called SHC. The protein SHC binds to the dimer, enabling a chain of protein–protein interactions inside the cell.

We model in LCLS the steps of the EGF pathway up to the binding of the protein SHC to the dimer. We model the EGFR protein as the sequence $R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}$, where R_{E1} and R_{E2} are two extra–cellular domains and R_{I1} and R_{I2} are two intra–cellular domains. The membrane of the cell is modeled as a looping sequence which could contain EGFR proteins. Outside the looping sequence (i.e. in the environment) there could be EGF proteins, and inside (i.e. in the cytoplasm) there could be SHC proteins. The rewrite rules modeling the pathway are the following:

$$EGF \mid \left(R_{E1} \cdot \widetilde{x}\right)^L \rfloor X \;\; \mapsto \;\; \left(SR_{E1} \cdot \widetilde{x}\right)^L \rfloor X \tag{R1}$$

$$\begin{aligned} &\left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto \\ &\quad \left(SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot \widetilde{y}\right)^L \rfloor X \end{aligned} \tag{R2}$$

$$\left(R^1_{E2} \cdot R_{I1} \cdot \widetilde{x} \cdot R^1_{E2} \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto \;\; \left(R^1_{E2} \cdot PR_{I1} \cdot \widetilde{x} \cdot R^1_{E2} \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \tag{R3}$$

$$\left(R^1_{E2} \cdot PR_{I1} \cdot \widetilde{x} \cdot R^1_{E2} \cdot R_{I1} \cdot \widetilde{y}\right)^L \rfloor X \;\; \mapsto \;\; \left(R^1_{E2} \cdot PR_{I1} \cdot \widetilde{x} \cdot R^1_{E2} \cdot PR_{I1} \cdot \widetilde{y}\right)^L \rfloor X \tag{R4}$$

$$\begin{aligned} &\left(R^1_{E2} \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{x} \cdot R^1_{E2} \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor (SHC \mid X) \;\; \mapsto \\ &\quad \left(R^1_{E2} \cdot PR_{I1} \cdot R^2_{I2} \cdot \widetilde{x} \cdot R^1_{E2} \cdot PR_{I1} \cdot R_{I2} \cdot \widetilde{y}\right)^L \rfloor (SHC^2 \mid X) \end{aligned} \tag{R5}$$

Rule R1 represents the binding of the EGF protein to the receptor domain R_{E1} with SR_{E1} as a result. Rule R2 represents that when two EGFR proteins activated by proteins EGF occur on the membrane, they may bind to each other to form a dimer (shown by the link 1). Rule R3 represents the phosphorylation of one of the internal domains R_{I1} of the dimer, and rule R4 represents the phosphorylation of the other internal domain R_{I1} of the dimer. The result of each phosphorylation is PR_{I1}. Rule R5 represents the binding of the protein SHC in the cytoplasm to an internal domain R_{I2} of the dimer. Remark that the binding of SHC to the dimer is represented by the link 2, allowing the protein SHC to continue the interactions to stimulate cell proliferation.

Let us denote the $R_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2}$ by EGFR. By starting from a cell with some EGFR proteins on its membrane, some SHC proteins in the cytoplasm and some EGF proteins in the environment, a possible evolution is the following (we write on each transition the name of the rewrite rule applied):

$$\begin{aligned} & EGF \mid EGF \mid \left(EGFR \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\ \xrightarrow{(R1)} \; & EGF \mid \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\ \xrightarrow{(R1)} \; & \left(SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot SR_{E1} \cdot R_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\ \xrightarrow{(R2)} \; & \left(SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \\ \xrightarrow{(R3)} \; & \left(SR_{E1} \cdot R^1_{E2} \cdot PR_{I1} \cdot R_{I2} \cdot SR_{E1} \cdot R^1_{E2} \cdot R_{I1} \cdot R_{I2} \cdot EGFR \cdot EGFR\right)^L \rfloor (SHC \mid SHC) \end{aligned}$$

$$\xrightarrow{(R4)} \quad (SR_{E1}\cdot R^1_{E2}\cdot PR_{I1}\cdot R_{I2}\cdot SR_{E1}\cdot R^1_{E2}\cdot PR_{I1}\cdot R_{I2}\cdot EGFR\cdot EGFR)^L \rfloor (SHC \mid SHC)$$

$$\xrightarrow{(R5)} \quad (SR_{E1}\cdot R^1_{E2}\cdot PR_{I1}\cdot R^2_{I2}\cdot SR_{E1}\cdot R^1_{E2}\cdot PR_{I1}\cdot R_{I2}\cdot EGFR\cdot EGFR)^L \rfloor (SHC^2 \mid SHC)$$

5 Conclusion

In previous papers we introduced the formalism Calculus of Looping Sequences (CLS) suitable to describe biological systems and their evolution.

In the present paper we have presented LCLS, an extension of CLS suitable to describe protein interaction at the domain level. A type system allows expressing well–formedness of terms and rewrite rules of the calculus, and an operational semantics is given which preserves well–formedness. We have shown an example of application of the calculus to the description of a classical biological system, namely the protein interactions of the EGF signalling pathway.

The relationship between CLS/LCLS and similar formalisms are studied in detail in [7]. Further work includes developing concepts of bisimulations for the new calculus in the line of what done for CLS.

References

1. R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin, and J. Schug. "Hybrid Modeling and Simulation of Biomolecular Networks". Hybrid Systems: Computation and Control, LNCS 2034, pages 19–32, Springer, 2001.
2. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. "A Calculus of Looping Sequences for Modelling Microbiological Systems". Fundamenta Informaticae, volume 72, number 1–3, pages 21–35, 2006.
3. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. "Bisimulation Congruences in the Calculus of Looping Sequences". Proc. of ICTAC'06, LNCS 4281, pages 93–107, 2006.
4. L. Cardelli. "Brane Calculi. Interactions of Biological Membranes". Proc. of CMSB'04, LNCS 3082, pages 257–280, Springer, 2005.
5. V. Danos and C. Laneve. "Formal Molecular Biology". Theoretical Computer Science, volume 325, number 1, pages 69–110, 2004.
6. C. Laneve and F. Tarissan. "A Simple Calculus for Proteins and Cells". Proc. of MeCBIC'06, ENTCS, to appear.
7. P. Milazzo. "Qualitative and Quantitative Formal Modeling of Biological Systems". PhD Thesis, University of Pisa, 2007.
8. H. Matsuno, A. Doi, M. Nagasaki, and S. Miyano."Hybrid Petri Net Representation of Gene Regulatory Network". Proc. of PSB'00,World Scientific Press, pages 341–352, 2000.
9. G. Păun. "Membrane Computing. An Introduction". Springer, 2002.
10. A. Regev, E.M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. "BioAmbients: An Abstraction for Biological Compartments". Theoretical Computer Science, volume 325, number 1, pages 141–167, 2004.
11. A. Regev, W. Silverman, and E.Y. Shapiro. "Representation and Simulation of Biochemical Processes Using the Pi-Calculus Process Algebra". Proc. of PSB'01, World Scientific Press, pages 459–470, 2001.

Author Index