Sérgio Campos (Ed.)

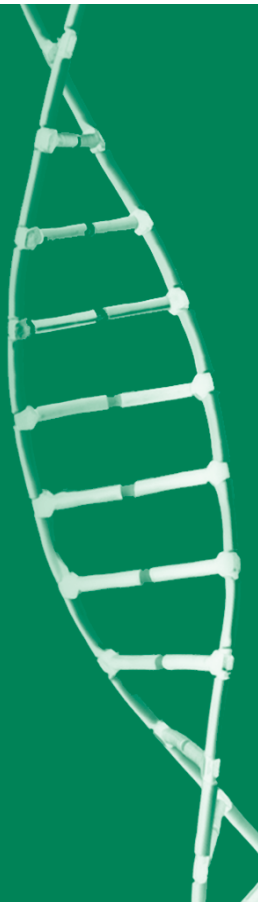
LNBI 8826

Advances in Bioinformatics and Computational Biology

9th Brazilian Symposium on Bioinformatics, BSB 2014
Belo Horizonte, Brazil, October 28–30, 2014
Proceedings

BRAZILIAN SYMPOSIUM ON BIOINFORMATICS

BSB 2014



Springer

Lecture Notes in Bioinformatics 8826

Subseries of Lecture Notes in Computer Science

LNBI Series Editors

LNBI Editorial Board

Sérgio Campos (Ed.)

Advances in Bioinformatics and Computational Biology

9th Brazilian Symposium on Bioinformatics, BSB 2014
Belo Horizonte, Brazil, October 28-30, 2014
Proceedings

Springer

Volume Editor

Sérgio Campos
Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Av. Antonio Carlos 6627
Pampulha, Belo Horizonte, MG, 31270-010, Brazil
E-mail: scampos@dcc.ufmg.br

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-319-12417-9 e-ISBN 978-3-319-12418-6
DOI 10.1007/978-3-319-12418-6
Springer Cham Heidelberg New York Dordrecht London

Library of Congress Control Number: 2014950925

LNCS Sublibrary: SL 8 – Bioinformatics

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

This volume contains papers selected for presentation at the 9^{th} Brazilian Symposium on Bioinformatics (BSB 2014), held in October, 2014, in Belo Horizonte, Minas Gerais, Brazil. BSB is an international conference that covers all aspects of bioinformatics and computational biology. This year the event was jointly organized by the special interest group in Computational Biology of the Brazilian Computer Society (Sociedade Brasileira de Computação — SBC), which has been the organizer of BSB for the past several years, and by the Brazilian Association for Bioinformatics and Computational Biology (AB^3C), which has been the organizer of the X-Meeting, another bioinformatics and computational biology event, also for several years. This year, the two events were jointly organized and were co-located, with the objective to promote the integration of both research communities. This was the second year that the events were co-located, the first being in 2013.

As in previous editions, BSB 2014 had an international Program Committee with 28 members. After a rigorous review process by the Program Committee, 18 papers were accepted to be presented at the event, and are printed in this volume.

BSB 2014 was made possible by the dedication and work of many people and organizations. We would like to express our sincere thanks to all Program Committee members as well as additional reviewers. Their names are listed in the pages that follow. We are also grateful to local organizers and volunteers for their valuable help and to the sponsors for making the event possible. Finally, we would like to thank all authors for their time and effort in submitting their work. It is for them and all other researchers that the event is organized, and without them it would not have a purpose.

This year, selected BSB papers were invited for a submission in an expanded format to a special issue of the BMC Bioinformatics journal. We thank BMC for this opportunity.

November 2014 Sérgio Campos

Organization

Conference Chairs

Sérgio Campos	Universidade Federal de Minas Gerais, Brazil
Guilherme Oliveira	Fundação Oswaldo Cruz, Brazil

Program Chair

Sérgio Campos	Universidade Federal de Minas Gerais, Brazil

Steering Committee

Ivan G. Costa	RWTH Aachen University, Germany
João Carlos Setubal	Universidade de São Paulo, Brazil
Luis Antonio Kowada	Universidade Federal Fluminense, Brazil
Marcelo de Macedo Brígido	Universidade de Brasilia, UnB, Brazil
Osmar Norberto de Souza	Pontifícia Universidade Católica, RS, Brazil
Ronnie Alves	Universidade Federal do Pará, Brazil

Program Committee

André Fujita	Universidade de São Paulo, Brazil
David Sankoff	University of Ottawa, Canada
Guilherme Telles	Universidade Estadual de Campinas, Brazil
Ivan G. Costa	RWTH Aachen University, Germany
João Carlos Setubal	Universidade de São Paulo, Brazil
Katia Guimarães	Universidade Federal de Pernambuco, Brazil
Marcelo Macedo Brígido	Universidade de Brasilia, UnB, Brazil
Maria Emilia T. Walter	Universidade de Brasilia, UnB, Brazil
Marie France Sagot	Inria, France
Marília D. V. Braga	INMETRO, Brazil
Nalvo Almeida Jr.	Universidade Federal de Mato Grosso do Sul, Brazil
Ronnie Alves	Universidade Federal do Pará, Brazil
Said S. Adi	Universidade Federal de Mato Grosso do Sul, Brazil
Sérgio Vale Aguiar Campos	Universidade Federal de Minas Gerais, Brazil

Additional Reviewers

Alberto Davila	Fundação Oswaldo Cruz, Brazil
Alessandra Faria-Campos	INMETRO, Brazil
Áurea Folgueras-Flatschart	INMETRO, Brazil

Claudio Leonardo Lucchesi	Universidade Federal de Mato Grosso do Sul, Brazil
Cristina Baracat Pereira	Universidade Federal de Viçosa, Brazil
Fabiana Goés	Universidade Federal do Pará, Brazil
Fábio Henrique Martinez	Universidade Federal de Mato Grosso do Sul, Brazil
Fernando Braz	Universidade Federal de Minas Gerais, Brazil
Francisco Elói de Araújo	Universidade Federal de Mato Grosso do Sul, Brazil
Jens Stoye	Bielefeld University, Germany
José Flávio Júnior	Universidade Federal do Pará, Brazil
Leandro Corrêa	Universidade Federal do Pará, Brazil
Mark Alan Junho Song	Pontifícia Universidade Católica, MG, Brazil

Sponsors

Sociedade Brasileira de Computação
Associação Brasileira de Bioinformática e Biologia Computacional

Intel Corporation

Funding

Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq
Coordenação de Aperfeiçoamento de Pessoal de Nível Superior, CAPES
Fundação de Amparo à Pesquisa do Estado de Minas Gerais, FAPEMIG
Intel Corporation

Table of Contents

An Extensible Framework for Genomic and Metagenomic Analysis

Luciano A. Digiampietri, Vivian M.Y. Pereira, Camila I. Costa, Geraldo J. dos Santos Júnior, Fernando M. Stefanini, and Caio R.N. Santiago

School of Arts, Sciences and Humanities, University of So Paulo (USP)
Av. Arlindo Bttio, Ermelino Matarazzo, 03828-000, So Paulo, SP, Brazil
digiampietri@usp.br

Abstract. Computational tools for supporting the management of scientific experiments are fundamental for the modern science. These tools must be easy to use, extensible and robust. This paper presents a framework for managing bioinformatics' experiments, focusing on analysis of genomic and metagenomic data. The developed system is based on an extension of a scientific workflows management system combined with the development of specific tools related to genomic and metagenomic data analysis.

Keywords: Scientific Workflow, Framework, Genomic and Metagenomic Analysis.

1 Introduction

The evolution of data acquisition equipment has revolutionized modern science [10]. It makes the use of computers increasingly imperative for the different areas of knowledge and demands the creation or expansion of systems to manage scientific experiments. These systems must be easy to use (because they will be used by scientists from different domains), extensible (to allow the addition of new features which is quite necessary due to the current dynamism of science), and robust (to enable data management and execution of experiments involving large volumes of data, and executions that can take days).

One area very tied to this evolution is bioinformatics - an essentially interdisciplinary field involving Biology, Computer Science, Statistics, Chemistry, Pharmacy, Mathematics, among others, for the development of methods for the storage and retrieval of biological data, and the construction of models and algorithms for the solution of biological problems.

In the early 90s it was very costly (in terms of time and money) sequencing a single bacterium genome, which typically has a few million base pairs [18]; but, nowadays, sequencers with high performance made possible to obtain large amounts of DNA (tens of millions of bases) in a single sequencing. This volume of data brought new challenges, one of the main concerns the fact that it is common in a single sequencing to obtain genetic material from individuals of thousands of different species. The purpose of such studies is to analyze the

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 1–8, 2014.

diversity of microbial life given a particular ecological niche (e.g. the stomach of an animal, or soil or water from specific locations). These projects are typically called environmental genomics or metagenomics projects.

Some of the main challenges related to metagenomics studies are: (i) the identification of sequences (reads) belonging to each species; (ii) genomes assembly, as there will likely be missing pieces (not allowing a complete assembly), and have repeated regions in each genome and along the different genomes; (iii) the analysis of the amount and diversity of organisms found; (iv) the verification if any of the sequenced species probably represents a new species (never previously sequenced); (v) the identification of active metabolic pathways; among others.

In order to help addressing some of these challenges, this paper presents the extension of a general purpose scientific workflow management system (SWMS) [20,8,15] adding new features to help the management of bioinformatics experiments. Moreover, we present some tools (workflows' activities) that were developed specifically for the management and analysis of genomic and metagenomic data. The system and the tools presented in this paper are contextualized in the "metagenomics of ecological niches of the Sao Paulo Zoo" project[1] [14] at the Genome Sciences Research Center at the University of Sao Paulo[2] .

The remainder of this paper is organized as follows. Section 2 summarizes the related work. Section 3 presents the developed system. Finally, Section 4 contains the conclusions and future work.

2 Related Work

Several scientific experiments and business processes involve the execution of a set of activities in a given order. It is increasingly common to find tools for performing each of these activities available as Web services, local applications, or libraries functions.

In the last years, several approaches have been proposed for storing, sharing, composing and executing scientific experiments as workflows. Some works concern about the automatic composition of activities, such as the ones based on the marriage of the services' interfaces, the use of semantics for the validation of data integration, and the use of Artificial Intelligence planning [11,17]. The majority of them are focused on the fully automatic composition of services, without providing specific tools for improving existing workflows or suggesting new activities to a user who is building her/his workflow [16].

Currently, there are some Scientific Workflows Management Systems (SWMS) available, such as: *Kepler* [2], *Triana* [19], *VisTrails* [5], and *Galaxy* [9]. All of them share some basic functionalities such as the composition of workflows using graphical interface. Moreover, in Kepler and Triana systems you can create components and workflow models (*templates*) that can be reused in other workflows. Detailed reviews on SWMSs can be found in [6,7,12].

[1] `http://www.iq.usp.br/setubal/metazoo.html`
[2] `http://www.iq.usp.br/napcg/`

Other works relevant to this paper are the ones that provide sets of tools for the analysis of genomic and metagenomic data. There are hundreds of tools available in different formats (Web services, local applications, scripts, classes, etc.) to support specific bioinformatics' tasks. Indeed, this wide availability was one of the motivations for the extension of the framework made in this paper, since the framework allows the use of classes in the Java programming language, local application (which may or may not be scripts), and Web services as basic workflows' activities (in an easy to use and transparent way). Thus, any of the tools available can be easily used in the framework. We highlight two initiatives for the provision of bioinformatics tools. The first is known as *bio**: BioPerl, BioJava, BioPython, among others [13] and corresponds to the availability of a set of open source bioinformatics tools in different programming languages. The second is a more recent initiative which is focused on metagenomes: *Community Bio-Perl toolkit* [4].

In terms of SWMS, the extensions made on this paper stand out by the easy incorporation of existing activities and the recommendation of activities to help the user during the workflows' construction (as will be detailed in the next section). In terms of specific bioinformatics' tools, this work aims to complement existing tools and to create interfaces between existing tools in order to simplify the end user's work.

3 Developed Framework

This section presents the developed system that aims to support the management and execution of bioinformatics experiments. This section is organized in two subsections: Workflows' Activities, which describes some of the tools implemented to serve as workflow activities, helping in genomic and metagenomic data analysis; and subsection Workflow Management System that summarizes the extensions that have been developed to enhance the framework used as a basis for scientific experiments management.

3.1 Workflows' Activities

In this project, several tools were developed to support the analysis of genomic and metagenomic related data, which are used as basic activities in the SWMS that will be presented in Subsection 3.2. Following, some of the main tools are briefly described, divided into two groups: general purpose and bioinformatics specific tools.

General Purpose Tools: Much of the data analyzed in bioinformatics are in text files, whose fields are delimited by a specific character (such as commas or tabs). Thus, general text manipulation tools were developed to support the operations on these files. These tools are quite simple, but its repeated use in different projects justified the creation of workflows of activities for them. They include

activities for selecting and filtering information from semi-structured file, operations on sets/lists (intersection, union and difference) and mathematics basic activities (for addition, subtraction, multiplication, division and exponentiation of numbers)[3].

Bioinformatics Tools: Three types of existing tools can be used as workflows' activities by the SWMS: methods written in the Java programming language, local applications (including scripts and executable programs), and Web services. In order to do this, it is only necessary the creation of an activity that encapsulates the call to these tools (details of the creation of these activities can be found in [8]). Thus, activities for the most common bioinformatics' tools (such as tools for local alignments, multiple alignments, generating phylogenetic trees, etc.) have already been created. In this subsection, we present only some of the tools that have been developed within this project[4].

Taxonomic Classification of Reads: This activity uses the results of reads alignments with NR[5] and the taxonomy provided by the NCBI for taxonomic classification. The differential of this tool when compared with similar tools for metagenomic reads' classification is the use of the full NR database for the classification (and not just some reference sequences); the user can parameterize the similarity required considering different parameters (coverage, score, evalue, alignment size, etc.), and there is the possibility of classifying a read using taxonomical higher levels whenever there is different classifications for the same read: for example, if a *read* meets the criteria of similarity with two or more sequences from different species then it will not be classified as belonging to any of these species, but to the taxonomical level shared by these species (genus, family, etc.).

Download of Set of Sequences Based on Their GI Identification: This activity performs the *download* of the nucleotide or aminoacid sequences based on its identifier. This tool is typically executed using as input a *blast* output file in order to obtain all the sequences related with one or more query sequences. These sequences can be used, for example, for performing multiple alignments, for creating phylogenetic trees or for creating graphs of relationships.

Web View of Multiple Alignments: This activity creates an HTML file with the result of a multiple sequence alignment. The results are similar to the ones produced by tools such as *seaview*, but they are in HTML format making easy the sharing and the selection of regions of interest.

Creation of Sequences' Relationship Graph: This activity is based on the results of local alignments or multiple alignments and creates a graph of relationships between sequences considering the similarity of sequences using

[3] A short description of these activities can be found in: `www.each.usp.br/digiampietri/BioWorkflows/index.html#basic`

[4] Screenshots and resulting images from the execution of some of these tools can be found in: `www.each.usp.br/digiampietri/BioWorkflows`

[5] Data set with non-redundant nucleotide sequences from NCBI: `http://www.ncbi.nlm.nih.gov/refseq/`

the parameters provided by the user. The connected components or the cliques from the resulting graph can be used to obtain a reference sequence in order to simplify phylogenetic analysis involving a large number of sequences.

Partial Order of Assemblies: This activity uses a genetic algorithm to pair the *contigs* from two different (partial) assemblies of the same species. This pairing is used to assist in the refinement of the assembly and production of *scaffolds*.;

Identification of Conflicts in Assemblies: This activity receives the *reads* from a sequencing and a region of the (partial) assembled genome and verify the amount of *reads* confirming the assembly of that region and/or conflicting with it.

Genome Assembly Excluding Conflicting Reads: Activity that receives a *reads* conflicting list (which can be generated by the previous activity), and the list of input fastq files and produces a new assembly excluding conflicting reads. This tool is typically used to exclude a small amount of reads that avoid the merge of two contigs by the assembler (in the current version, this activity uses Newbler[6] as assembler).

3.2 Workflow Management System

In this paper we extend a SWMS originally designed in 1999 [15] and whose details about the last version of the system (before the extension made in this paper) can be found in [8][7]. There is a workflows editor where each activity can be a Web service, a method in Java programming language, or a local application. The graphical interface allows the creation, edition and execution of workflows (scientific experiments), as well as the conversion of a workflow into an executable code, which can then be executed outside the developed environment. By being a general purpose SWMS, the framework is used by different areas of knowledge (such as in the processing of biological images [3]).

Figure 1 presents the framework's architecture. The modules that were added or extended in this paper are highlighted.

A workflow is composed of *activities* (rectangles in the graphical representation), each activity can have a set of inputs and outputs; *data flows* (black arrows) that connect an output from one activity to the input of another; and *control flow* (gray arrows) indicating that an activity can only be started after the execution of another.

Figure 2 contains the screenshot of a workflow that is responsible for running the *blast* program, aligning the sequences from the input file with a database of sequences (in the example, we used the NR database from NCBI); the sequences that align with the input sequence are then downloaded from NCBI, a multiple alignment is produced, and, based on this multiple alignment, a similarity graph is built[8]; and a phylogenetic tree.

[6] `http://en.wikipedia.org/wiki/Newbler`

[7] All functionalities from the previous version of the system were kept in the expansion and some new ones were added as will be presented.

[8] This graph can be found in: `www.each.usp.br/digiampietri/BioWorkflows`

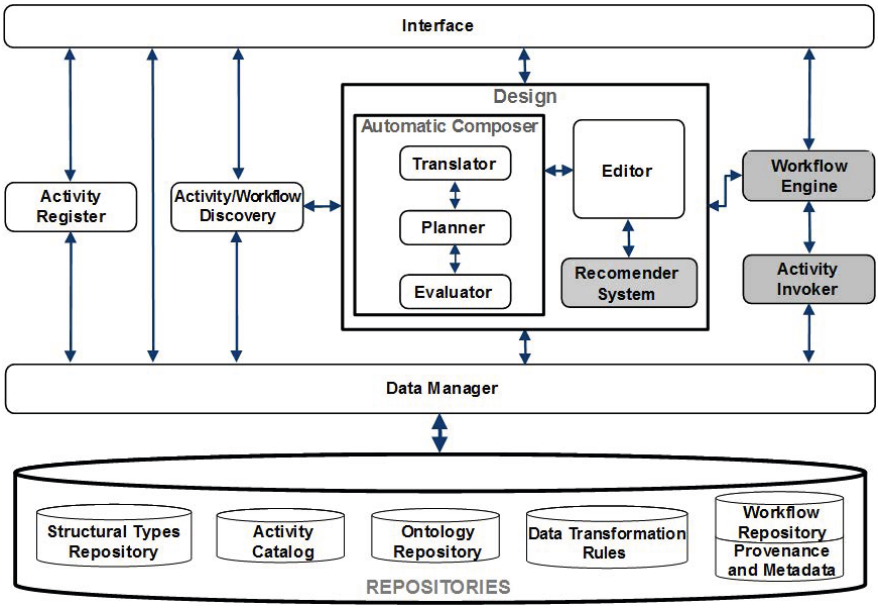


Fig. 1. SWMS Architecture

Due to space limitations, only the two main new features of the SWMS will be presented. They aim to improve the previous version of the SWMS in order to address some bioinformatics experiments' needs: improvement of the **workflows execution model** and development of an **activity recommender system**.

The **workflow execution model** was modified to allow a choreographed execution of workflows. In the previous version, the framework allowed only an orchestrated execution, restricting the creation of loops and conditionals, and did not allow the execution of workflows in a distributed way using, for example, volunteer computing. Thus, this modification was developed to enhance the performance of the distributed execution of bioinformatics experiments.

As stated in Section 2, there are several approaches to assist the end user in the construction of workflows. The **activity recommender system** was developed to provide a semi-automatic composition approach, since the automatic composition system already exists in the current system, which best fits users with no knowledge of the application domain; while semi-automatic approaches aim to help in the construction of workflows without imposing an outcome, ensuring the autonomy of the user. Such systems suggest activities for the user which can be used in the workflow to complete its construction. For this, two approaches were used: the use of the *Apriori* algorithm [1] to generate association rules, i.e., rules that say *if the workflow has the activities* x *and* y *then it probably will have the activity* z (in this case the system will recommends to the user the inclusion of activity z). This approach has been used in workflow systems [16], but it has one limitation: the calculation of the rules is quite slow due to the large amount of activities available. To deal with this limitation a

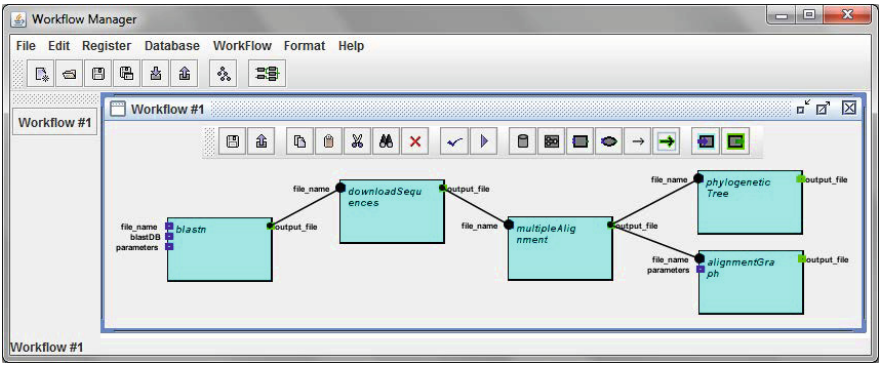


Fig. 2. Screenshot of the SWMS

second recommendation approach was developed: for each activity a descending ordered list of the most frequent activities that usually occur after this activity is created and whenever a user inserts a new activity in the workflow, the system suggests what are the next most potentially useful activities based on this list. In order to create the rules and the lists of activities, as well as to validate the approaches we used the workflows available in *MyExperiment*[9].

4 Conclusion

The development of systems to support the management and execution of scientific experiments has become increasingly important. This paper presented a bioinformatics SWMS, focusing on genomic and metagenomic data analysis. The developed system is based on an extension of a SWMS combined with the development of specific activities related to genomic and metagenomic data analysis.

As future work we intend to develop new tools for the analysis of metagenomic data, and evaluate the performance improvement of using the distributed execution of workflows with volunteer computing.

Acknowledgments. This work was partially funded by FAPESP, CNPq, Tutorial Education Program of Brazilian Ministry Education (PET/MEC), and University of Sao Paulo.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: International Conference On Very Large Data Bases (VLDB 1994), pp. 487–499. Morgan Kaufmann Publishers, Inc. (1994)

[9] `http://www.myexperiment.org/`

2. Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S.: Kepler: An extensible system for design and execution of scientific workflows. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management, Washington, DC, USA, pp. 423–424 (2004)
3. Andrioli, L.P., Digiampietri, L.A., de Barros, L.P., Machado-Lima, A.: Huckebein is part of a combinatorial repression code in the anterior blastoderm. Developmental Biology 361(1), 177–185 (2012)
4. Angly, F.E., Fields, C.J., Tyson, G.W.: The bio-community perl toolkit for microbial ecology. Bioinformatics (2014)
5. Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva, C.T., Vo, H.T.: Managing the evolution of dataflows with VisTrails. In: Proceedings of the 22nd International Conference on Data Engineering Workshops, p. 71 (2006)
6. Curcin, V., Ghanem, M.: Scientific workflow systems - can one size fit all? In: Biomedical Engineering Conference, CIBEC 2008, pp. 1–9 (2008)
7. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528–540 (2009)
8. Digiampietri, L.A., Araujo, J.C., Ostroski, E.H., Santiago, C.R.N., Perez-Alcazar, J.J.: Combinando workflows e semantica para facilitar o reuso de software. Revista de Informatica Teorica e Aplicada: RITA 20, 73–89 (2013)
9. Goecks, J., Nekrutenko, A., Taylor, J., Team, T.G.: Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biology 11(8), R86 (2010)
10. Hey, T., Tansley, S., Tolle, K. (eds.): The Fourth Paradigm: Data-Intensive Scientific Discovery. Microsoft Research, Redmond (2009)
11. Long, D., Fox, M.: The 3rd International Planning Competition: Results and Analysis. Journal of Artificial Intelligence Research 20, 1–59 (2003)
12. Lu, S., Zhang, J.: Collaborative scientific workflows. In: IEEE International Conference on Web Services, ICWS 2009, pp. 527–534 (2009)
13. Mangalam, H.: The bio* toolkits - A brief preview. Briefings in Bioinformatics 3(3), 296–302 (2002)
14. Martins, L.F., Antunes, L.P., Pascon, R.C., de Oliveira, J.C.F., Digiampietri, L.A., Barbosa, D., Peixoto, B.M., Vallim, M.A., Viana-Niero, C., Ostroski, E.H., et al.: Metagenomic Analysis of a Tropical Composting Operation at the São Paulo Zoo Park Reveals Diversity of Biomass Degradation Functions and Organisms. Plos One 8(4), e61928 (2013)
15. Medeiros, C., Perez-Alcazar, J., Digiampietri, L., Pastorello, G., Santanche, A., Torres, R., Madeira, E., Bacarin, E.: WOODSS and the Web: Annotating and Reusing Scientific Workflows. ACM SIGMOD Record 34(3), 18–23 (2005)
16. Oliveira, F.T.: Um sistema de recomendacao para composicao de workflows. Master's thesis, COPPE/UFRJ (2010)
17. Rao, J., Su, X.: A survey of automated web service composition methods. In: Cardoso, J., Sheth, A.P. (eds.) SWSWPC 2004. LNCS, vol. 3387, pp. 43–54. Springer, Heidelberg (2005)
18. Setubal, J., Meidanis, J.: Introduction to Computational Molecular Biology. PWS Publishing Company, Boston (1997)
19. Taylor, I., Shields, M., Wang, I., Harrison, A.: Visual grid workflow in triana. Journal of Grid Computing 3(3-4), 153–169 (2005)
20. Zuniga, J.C., PrezAlczar, J.J., Digiampietri, L.A., Barbin, S.E.: A loosely coupled architecture for automatic composition of web services applications. International Journal of Metadata, Semantics and Ontologies 9(3), 241–251 (2014)

On the Multichromosomal Hultman Number

Pedro Feijão[1,2], Fábio Viduani Martinez[1,2,3], and Annelyse Thévenin[1,2]

[1] Genome Informatics, Faculty of Technology, Bielefeld University, Germany
[2] Technische Fakultät and CeBiTec, Universität Bielefeld, Germany
[3] Faculdade de Computação, Universidade Federal de Mato Grosso do Sul, Brazil

Abstract. The number of cycles of a breakpoint graph is one of the notable parameters to solve distance problems in comparative genomics. For a fixed c, the number of linear unichromosomal genomes with n genes such that the breakpoint graph has c disjoint cycles, the *Hultman number*, is already determined. In this work we extend this result to multichromosomal genomes, providing formulas to compute the number of multichromosal genomes having a fixed number of cycles and/or paths.

1 Introduction

In molecular biology, several measures have been proposed to compute the (dis)similarity between genomes. In *genome rearrangements*, one is concerned with measures of dissimilarity involving large-scale mutations, where a fundamental problem is to determine the smallest sequence of such operations that transforms one given genome into another. This minimum number of operations is called the *rearrangement distance* between the two given genomes (see [5]).

A remarkable characteristic of methods to compute distances is the systematic use of a graph, first introduced by Bafna and Pevzner [2], known as the *breakpoint graph*. It has proven, by its decomposition into disjoint cycles, a useful tool to efficiently compute rearrangement distances such as transposition or reversal, directly related to the number of cycles in this decomposition [5].

Since cycle decomposition of breakpoint graphs plays a central role in computing distances, it is useful to investigate the distribution of such cycles. Particularly, the distribution of genomes with a number of cycles c allows us to evaluate the probability to have a scenario of a distance d depending of c. Doignon and Labarre [4] enumerated the unsigned permutations of a given size such that the corresponding graph has a given number of cycles, and called it the *Hultman number*. Subsequently, Grusea and Labarre [7] extended this result for signed permutations, which properly represent unichromosomal genomes. In this work we extend previous results providing formulas to compute the number of multichromosomal genomes with a given number of cycles and/or paths. We obtain an explicit formula for circular genomes and recurrences for more general cases.

In Section 2 of this work we introduce definitions, notations, and details of previous results. Results for circular and general multichromosomal genomes are presented in Section 3, and Section 4 presents conclusions and perspectives.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 9–16, 2014.

2 Preliminaries

We represent multichromosomal genomes using a similar notation as in [3]. A *gene* is a fragment of DNA on one of the two DNA strands in a chromosome, showing its orientation. A gene is represented by an integer and its orientation by a sign. The orientation of a gene g allows us to distinguish its two *extremities*, the *tail* (g^t) and the *head* (g^h). A *chromosome* is represented by a sequence of genes, flanked in the extremities by *telomeres* ($\circ$) if the chromosome is linear; otherwise, it is circular. *Genomes* are represented as sets of chromosomes. An *adjacency* in a genome is either a pair of consecutive gene extremities in a chromosome, or a gene extremity adjacent to a telomere (a *telomeric adjacency*). For instance, $A = \{(\circ\ 1\ 2\ 3\ 4\ \circ)\}$ is a genome with one linear chromosome and four genes, and has the adjacencies $\circ 1^t, 1^h 2^t, 2^h 3^t, 3^h 4^t$ and $4^h \circ$, where the first and the last are telomeric adjacencies.

There is a one-to-one correspondence between genomes and *matchings* in the set of extremities. Adjacencies correspond to two matched (saturated) vertices, and telomeric adjacencies correspond to unmatched (unsaturated) vertices. Therefore, a perfect matching (i.e., matching which saturates all vertices of the graph) corresponds to a genome with only circular chromosomes. The matching corresponding to a genome A is denoted by M_A. Because of this one-to-one relationship, in this text we use the terms *genome* and *matching* interchangeably.

Given two genomes A and B with the same set of genes, the *multichromosomal breakpoint graph* of A and B, denoted by $BG(A, B)$, is built by joining the matchings M_A and M_B in the same set of vertices, using different colors for the edges of each matching. Figure 1 shows an example of a multichromosomal breakpoint graph for genomes $A = \{(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)\}$ and $B = \{(6\ -1\ 4\ 5\ -2), (\circ\ -9\ 3\ 8\ 7\ \circ)\}$. From this point on we will use the term *breakpoint graph* to refer to the multichromosomal breakpoint graph. Since all its vertices have degree 0, 1 or 2, the breakpoint graph is uniquely decomposed in cycles and paths. For instance, the breakpoint graph in Figure 1 is decomposed in two cycles and one path.

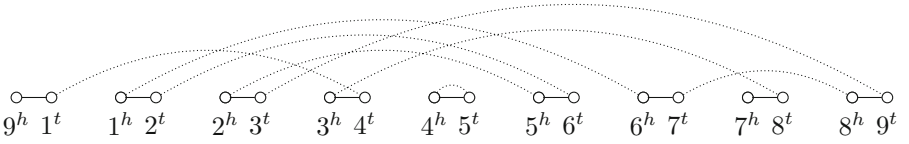


Fig. 1. Multichromosomal breakpoint graph for genomes $A = \{(1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9)\}$ (black edges) and $B = \{(6\ -1\ 4\ 5\ -2), (\circ\ -9\ 3\ 8\ 7\ \circ)\}$ (dotted edges)

3 The Multichromosomal Hultman Number

In this section, we extend the results of [4,7] for multichromosomal genomes. There are two new aspects that must be considered. First, since the breakpoint

graph can be decomposed in cycles and paths, we may have to count not only cycles, but also paths. The other question is about the *identity genome*. In the unichromosomal case, the identity genome is easily defined. In the multichromosomal case, it is not obvious which given genome is the identity. When working on multichromosomal circular genomes (Section 3.1), the identity is defined as in the unichromosomal case. In the general case, working on genomes with linear and circular chromosomes (Section 3.2), we analyse two types of identities for genomes: one with only one set of circular chromosomes and another with a set of circular chromosomes and a set of linear chromosomes.

In the next subsections, we propose extensions of the Hultman number for multichromosomal genomes, first considering only circular genomes, and then extending the results to general genomes, with linear and circular chromosomes. The same strategy is used in all cases: first, start with a matching representing the identity, and then superimpose all other possible matchings, while counting recursively cycles and paths. To do that, we need to consider all possible operations to build such matchings. In Figure 2, all such operations are shown.

3.1 Circular Genomes

A *circular genome* is a genome where all chromosomes are circular. Since there are no telomeric adjacencies, the matching M_A of a circular genome A is a perfect matching on the extremities of A. The breakpoint graph of two circular genomes is decomposed in disjoint alternating cycles, since each vertex has degree two.

We compute the number of circular genomes with n genes that have c disjoint alternating cycles over a given identity genome I, and call it the *multichromosomal circular Hultman number*, denoted by $H_C(n, c)$. In this case, since the matching of any circular genome is a perfect matching, we claim that $H_C(n, c)$ is the same, independently of the genome I chosen as an identity, and simply define $I_\circ = \{(1, 2, \ldots, n)\}$. Hence, we define $H_C(n, c) = |\{A \in \mathcal{C}_n : cyc(BG(A, I_\circ)) = c\}|$, where $\mathcal{C}_n$ is the set of all circular multichromosomal genomes with n genes and $cyc(\cdot)$ denotes the number of cycles in a graph.

Starting with a perfect matching $M_{I_\circ}$ of the $2n$ vertices, we build all breakpoint graphs $BG(A, I_\circ)$, for circular genomes A, which correspond to perfect matchings, adding one edge at a time, while counting the number of cycles, recursively. The matching $M_{I_\circ}$ is composed by n (connected) components, and all are paths. Considering an arbitrary vertex u in the matching $M_{I_\circ}$, there are $2n - 1$ possible edges uv that can be created. Figure 2(a,b) shows how these different edges can be chosen. There are two possible cases:

(a) Create Cycle: If u and v belong to the same component, uv will *create a cycle*. There is only one possibility for this type of edge.

(b) Merge Paths: If u and v belong to different components, uv will *merge both paths*. There are $2n - 2$ possibilities of adding such an edge.

We establish a recurrence for $H_C(n, c)$, where $0 \leq c \leq n$, as follows:

$$H_C(n, c) = \begin{cases} 0\,, & \text{if } n < c \text{ or } c = 0\,, \\ 1\,, & \text{if } n = c\,, \\ H_C(n-1, c-1) + (2n-2) \cdot H_C(n-1, c)\,, & \text{if } n > c\,. \end{cases}$$

Since $1 \leq c = cyc(BG(A, I_\circ)) \leq n$ for every pair of genomes $A, I_\circ$ with n genes, then the first base case holds. There exists only one configuration where $c = cyc(BG(A, I_\circ)) = n$, when $A = I_\circ$, and thus the second base case holds.

Theorem 1. *There exists an explicit formula to $H_C(n, c)$:*

$$H_C(n,c) = \frac{2^{n-c}}{(c-1)!} \sum_{\substack{0 \leq q_1, \ldots, q_{n-c}: \\ \sum_2^{n-c} m q_m = n-c}} \frac{(n+Q-1)!}{q_2! \cdots q_{n-c}!\ 1!^{q_1} 2!^{q_2} \cdots k!^{q_{n-c}}},$$

where $Q = q_2 + \ldots + q_k$ and $\sum_2^{n-c} m q_m = n - c$ is a sum over all partitions of $n - c$.

Proof. We know from [6] that unsigned Stirling numbers of first kind satisfy the following recurrence equation: $\left[{n \atop c}\right] = \left[{n-1 \atop c-1}\right] + (n-1)\left[{n-1 \atop c}\right]$. Multiplying both sides by 2^{n-c} and using $H_C(n, c)$ recurrence equation we arrive at $H_C(n, c) = 2^{n-c}\left[{n \atop c}\right]$. Then, using the explicit formula for $\left[{n \atop c}\right]$ given in [9], we arrive at our result. □

Furthermore, the sequence of integers generated by $H_C(n, c)$ is the unsigned entry A039683 in the OEIS (On-Line Encyclopedia of Integer Sequences) [10].

3.2 General Genomes

We will generalize our previous formula for general multichromosomal genomes, with both linear and circular genomes. Now, we have not only cycles but also paths in the breakpoint graph. Thus, it is not clear which genome should be considered the identity genome. As a starting point, let us consider again the identity as $I_\circ = \{(1, 2, \ldots, n)\}$, and find the *general Hultman number* $H_G(n, c, p)$, defined as $H_G(n, c, p) = |\{A \in \mathcal{G}_n : cyc(BG(A, I_\circ)) = c \text{ and } path(BG(A, I_\circ)) = p\}|$, where $\mathcal{G}_n$ is the set of all multichromosomal genomes with n genes, and $path(\cdot)$ denotes the number of paths in a graph. In this set, each genome corresponds to a matching, not necessarily perfect, since only circular genomes correspond to perfect matchings. Similarly as the previous case, we start with the matching $M_{I_\circ}$ on $2n$ vertices, and recursively build all possible matchings, while counting cycles and paths. Since a matching induced by an arbitrary genome A in $\mathcal{G}_n$ is not necessarily perfect, together with the *create cycle* and *merge paths* operations on a vertex u (Section 3.1), we can also choose not to saturate a vertex u in the matching, creating a telomeric adjacency, which we call a *vertex skip* operation.

Moreover, since we now have an operation that is applied on just one vertex, and not two at a time such as the operations presented in Section 3.1, we need to define a different recurrence, where n correspond to vertices in the breakpoint graph, and not to genes in the genomes. In a genome $I_\circ$ with n genes, there are $2n$ vertices (extremities) in $M_{I_\circ}$ and consequently in $BG(A, I_\circ)$. So, we need an auxiliary number $H'_G(n', c, p)$, such that $H_G(n, c, p) = H'_G(n', c, p)$, with $n' = 2n$, and $H'_G(n', c, p) = |\{M \in \mathcal{M}_{n'} : cyc(BG(M, M_{I_\circ})) =$

c and $path(BG(M, M_{I_\circ})) = p\}|$, where $\mathcal{M}_{n'}$ is the set of all possible matchings on n' vertices, and $M_{I_\circ}$ is a perfect matching with $n'/2$ edges induced by $I_\circ$.

Starting with the matching $M_{I_\circ}$, another matching is built recursively by adding edges or skipping vertices until all vertices have been *visited*. Visited vertices are shown in figures as black vertices, and *unvisited* as white. If n' is even, we pick any unvisited vertex u and we have tree possibilities (Fig. 2 a,b,c):

(a) Create Cycle: There is one edge uv such that $v(\neq u)$ is the unvisited vertex in the same component as u, and this edge (shown as a grey edge uv) will create a cycle. Vertices u and v are marked as visited.

(b) Merge Paths: There are $n' - 2$ edges uv such that v is an unvisited vertex in a different component as u, and this edge will merge these components, that are paths. Vertices u and v are marked as visited.

(c) Skip Vertex: no edge is created and u is marked as visited.

If n' is odd, there is a vertex u that is connected to a visited vertex. From this vertex there is no way to create a cycle, but two operations are possible (Fig. 2e, f):

(e) Merge Paths: There are $n' - 1$ edges uv such that v is in a different component as u, merging these components. Vertices u and v are marked as visited.

(f) Skip Vertex: no edge is created, u is marked as visited. A path where all vertices are visited is created.

For the base cases, we notice that there is no possible matching if $n' \leq 0$. Also, since each cycle has at least two vertices and a path has at least one, no matching has $2c + p > n'$. For $n' = 1$, only one matching is possible, with one path and no cycle, therefore $H'_G(1, 0, 1) = 1$. For $n' = 2$, there is only one matching, with one cycle and no path, therefore $H'_G(2, 1, 0) = 1$. With that, we arrive at the following recurrence:

$$H'_G(n', c, p) = \begin{cases} 0, & (1) \\ 1, & (2) \\ H'_G(n' - 2, c - 1, p) + (n - 2) \cdot H'_G(n' - 2, c, p) + H'_G(n' - 1, c, p), & (3) \\ (n - 1) \cdot H'_G(n' - 2, c, p) + H'_G(n' - 1, c, p - 1), & (4) \end{cases}$$

with (1) if $n' \leq 0$ or $2c+p > n'$, (2) if $n' = 1, c = 0, p = 1$ or $n' = 2, c = 1, p = 0$, (3) if n' is even, and (4) if n' is odd.

3.3 General Genomes with a Fixed Number of Linear Chromosomes

In this section we generalize the previous approach for different identity genomes. Instead of fixing the identity as a circular genome, the identity I_ℓ is a genome with a fixed number of ℓ linear chromosomes. We define the Hultman number $H_L(n, c, p, \ell) = |\{A \in \mathcal{G}_n : cyc(BG(A, I_\ell)) = c \text{ and } path(BG(A, I_\ell)) = p\}|$, where $\mathcal{G}_n$ is the set of all multichromosomal genomes with n genes, and I_ℓ is a genome with exactly ℓ linear chromosomes. This is a generalization of the previous case, since $H_G(n, c, p) = H_L(n, c, p, 0)$.

We propose again an auxiliary series, defined as $H'_L(n', c, p, \ell') = |\{M \in \mathcal{M}_n : cyc(BG(M, M_{I_{\ell'}})) = c \text{ and } path(BG(M, M_{I_{\ell'}})) = p\}|$, where $\mathcal{M}_n$ is the set of all possible matchings on n' vertices, and $M_{I_{\ell'}}$ is a matching on these vertices such that exactly ℓ' vertices are unsaturated (isolated), with $n' = 2n$ and $\ell' = 2\ell$. Then, given a matching $M_{I_{\ell'}}$ with ℓ' unsaturated vertices, we will build a matching recursively adding edges or skipping vertices until all vertices have been visited. In this case, the parity of $n' + \ell'$ determines which possibilities we have (Figure 2). When $n' + \ell'$ is even, we will call the current state *balanced*, otherwise it is *unbalanced*. In the balanced case, focusing on an unvisited vertex u that is saturated by $M_{I_{\ell'}}$ there are four possible cases (Fig. 2a–d):

(a) Create Cycle: There is one edge uv such that $v(\neq u)$ is an unvisited vertex in the same component as u, and this edge will create a cycle. Vertices u and v are marked as visited.
(b) Merge Paths: There are $n' - 2 - \ell'$ edges uv such that v is saturated in $I_{\ell'}$ and is in a different component as u, and uv will merge these components, that are paths. Vertices u and v are marked as visited.
(c) Skip Vertex: No edge is created and u is marked as visited.
(d) Connect with Unsaturated: There are ℓ' possible edges from u to an unsaturated vertex v in $I_{\ell'}$. Vertices u and v are marked as visited.

Cases (a) and (b) visit two vertices that are saturated in $I_{\ell'}$, which means that the state remains balanced. Case (c) changes the state to unbalanced, since only one vertex is visited. Case (d) visits two vertices, but one is a unsaturated vertex in $I_{\ell'}$, which means that the parity of $n' + \ell'$ changes and the state becomes unbalanced.

In the unbalanced state, focusing on a vertex u belonging to a component with all other vertices visited, there are three possibilities (Fig. 2e–g):

(e) Merge Paths: There are $n' - 1 - \ell'$ edges uv such that v is saturated in $I_{\ell'}$ and is in a different component as u, and this edge will merge these components, that are paths. Vertices u and v are marked as visited.
(f) Skip Vertex: Vertex u is not saturated in M; no edge is created and only u is marked as visited, and a path with all vertices visited is created.
(g) Connect with Unsaturated: There are ℓ' possible edges from u to an unsaturated vertex v in $I_{\ell'}$. Vertices u and v are marked as visited, and a path with all vertices visited is created.

Cases (e), (f) and (g) are similar to cases (b), (c) and (d), respectively, which means that (e) keeps the state unbalanced, but (f) and (g) change it to balanced again. There are still two cases to consider, when $n' = \ell'$ (Fig. 2h, i).

(h) Connect Two Unsaturated: There are $\ell' - 1$ possible edges from an unsaturated vertex u to an unsaturated vertex v in $I_{\ell'}$. Vertices u and v are marked as visited, and a path with all vertices visited is created.
(i) Skip Vertex: No edge is created and u is marked as visited. A path with all vertices visited is created.

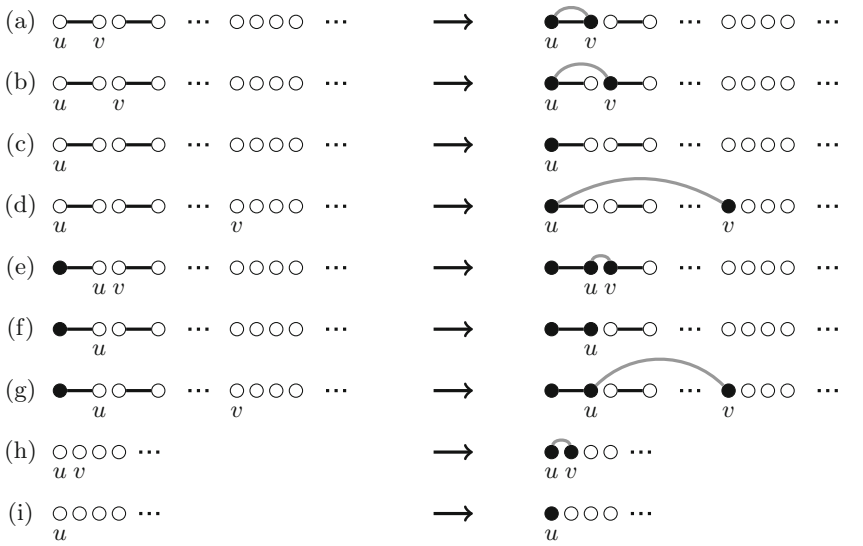


Fig. 2. Operations for recursively building a matching (new genome) on top of an existing matching (identity genome), resulting in a breakpoint graph. Unvisited vertices are white, visited are black. From vertex u we have the following operations: *Balanced configurations*: (a) Create cycle, (b) Merge paths, (c) Skip vertex, (d) Connect unsaturated; *Unbalanced configurations*: (e) Merge paths, (f) Skip vertex, (g) Connect unsaturated, (h) Connect two unsaturated, (i) Skip unsaturated.

For the base cases, as before there is no possible matching when $n' \leq 0$ or $2c+p > n'$. For $n' = 1$, only one matching is possible with one path. For $n' = 2$, there is only one matching with one path and no cycle or with one cycle and no path. With all these cases described, we arrive at the recurrence:

$$H'_L(n', c, p, \ell') = \begin{cases} 0, & (1) \\ 1, & (2) \\ (\ell' - 1) \cdot H'_L(n' - 2, c, p - 1, \ell' - 2) + H'_L(n' - 1, c, p - 1, \ell' - 1), & (3) \\ H'_L(n' - 2, c - 1, p, \ell') + (n' - 2 - \ell') \cdot H'_L(n' - 2, c, p, \ell') + \\ \qquad i \cdot H'_L(n' - 2, c, p, \ell' - 1) + H'_L(n' - 1, c, p, \ell'), & (4) \\ (n' - 1 - \ell') \cdot H'_L(n' - 2, c, p, \ell') + \\ \qquad i \cdot H'_L(n' - 2, c, p - 1, \ell' - 1) \ + H'_L(n' - 1, c, p - 1, \ell'), & (5) \end{cases}$$

with (1) if $n' \leq 0$ or $2c + p > n'$, (2) if $n' = p = 1, c = 0$ or $n' = 2, c = 0, p = 1$ or $n' = 2, c = 1, p = \ell' = 0$, (3) if $n' = \ell'$, (4) if $n' + \ell'$ is even, $n' > \ell'$, (5) if $n' + \ell'$ is odd, $n' > \ell'$.

4 Conclusion

In this paper, we introduced different recurrence equations for the Hultman number and its variations, that are relevant in the context of comparative genomics. We have extended previous results that treated the unichromosomal cases [4,7], focusing on multichromosomal genomes. Table 1 shows a summary of the results.

Table 1. Summary of the results in this paper. The first two lines show previous results, and the last three the Hultman numbers proposed in this paper.

Hultman Number	**Identity**	**Universe**
$\mathcal{H}(n,k)$ [4]	$\pi = \langle 1 \ \cdots \ n \rangle$	S_n (unsigned permutations)
$\mathcal{H}^{\pm}(n,k)$ [7]	$\pi = \langle 1 \ \cdots \ n \rangle$	$S_n^{\pm}$ (signed permutations)
$H_C(n,c)$	Circular Genome	Circular genomes
$H_G(n,c,p)$	Circular Genome	General genomes
$H_L(n,c,p,\ell)$	Genome with ℓ linear chr.	General genomes

For the circular Hultman number $H_C(n,c)$ we also provided an explicit formula, using the relationship between this series and the unsigned Stirling numbers of first kind. Future directions include finding explicit equations for the introduced recursive equations and finding new Hultman numbers for other relevant biological comparisons, such as restricting genomes to have the same number of linear and/or circular chromosomes.

References

1. Bafna, V., Pevzner, P.P.A.: Sorting by transpositions. SIAM Journal on Discrete Mathematics 11(2), 224–240 (1998)
2. Bafna, V., Pevzner, P.A.: Genome rearrangements and sorting by reversals. SIAM Journal on Computing 25(2), 272–289 (1996)
3. Bergeron, A., Mixtacki, J., Stoye, J.: A unifying view of genome rearrangements. In: Bücher, P., Moret, B.M.E. (eds.) WABI 2006. LNCS (LNBI), vol. 4175, pp. 163–173. Springer, Heidelberg (2006)
4. Doignon, J., Labarre, A.: On Hultman numbers. Journal of Integer Sequences 10(6), article 07.6.2, 13 p. (2007)
5. Fertin, G., Labarre, A., Rusu, I., Tannier, E., Vialette, S.: Combinatorics of Genome Rearrangements. MIT Press (2009)
6. Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete Mathematics: A Foundation for Computer Science. Addison Wesley (1994)
7. Grusea, S., Labarre, A.: The distribution of cycles in breakpoint graphs of signed permutations. Discrete Applied Mathematics 161(10-11), 1448–1466 (2013)
8. Hultman, A.: Toric permutations. Master's thesis, Department of Mathematics. KTH, Stockholm, Sweden (1999)
9. Malenfant, J.: Finite, closed-form expressions for the partition function and for Euler, Bernoulli, and Stirling numbers. ArXiv e-prints (March 2011)
10. Sloane, N.J.A.: The On-line Encyclopedia of Integer Sequences – OEIS (2014), `http://oeis.org`

Towards an Ensemble Learning Strategy for Metagenomic Gene Prediction

Fabiana Goés[1], Ronnie Alves[1,2,4], Leandro Corrêa[1], Cristian Chaparro, and Lucinéia Thom[3]

[1] PPGCC, Universidade Federal do Pará, Belém, Brazil
fabii.goes@gmail.com
[2] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, UMR 5506, Université Montpellier 2,
Centre National de la Recherche Scientifique, Montpellier, France
alvesrco@gmail.com
[3] PPGC, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
lucineia@inf.ufrgs.br
[4] Institut de Biologie Computationnelle, Montpellier, France

Abstract. Metagenomics is an emerging field in which the power of genome analysis is applied to entire communities of microbes. A large variety of classifiers has been developed for gene prediction though there is lack of an empirical evaluation regarding the core machine learning techniques implemented in these tools. In this work we present an empirical performance evaluation of classification strategies for metagenomic gene prediction. This comparison takes into account distinct supervised learning strategies: one lazy learner, two eager-learners and one ensemble learner. Though the performance of the four base classifiers was good, the ensemble-based strategy with *Random Forest* has achieved the overall best result.

Keywords: Machine learning, classification methods, gene prediction, metagenomics.

1 Introduction

Metagenomics is an emerging field in which the power of genome analysis is applied to entire communities of microbes, bypassing the need to isolate and culture individual microbial species [1]. It is focused on the understanding of the mixture of genes (genomes) in a community as a whole. The gene prediction task is a well-known problem in genomics, and it remains an interesting computational challenge in metagenomics as well. Depending on the applicability and success of the assembly, gene prediction can be done on post assembly *contigs*[1], on reads from unassembled metagenomes or on a mixture of *contigs* and individual unassembled reads. There are two main strategies for gene prediction [2]: i) **evidence-based** gene-calling methods use homology searches to

[1] A contig is a continuous sequence resulting from the assembly of overlapping small DNA fragments (sequence reads).

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 17–24, 2014.

find genes similar to those observed previously (reference microbial genomes); and ii) **ab initio** gene-calling relies on the intrinsic features of the DNA sequence to discriminate between coding and noncoding regions, allowing for the identification of homologs in the available databases. The former approach has two major drawbacks. Low values of similarity to known sequences either due to evolutionary distance or due to the short length of metagenomic coding sequences and the presence of sequence errors restrict the identification of homologs. In addition, novel genes without similarities are completely ignored. The latter approach usually employs Machine Learning (ML) algorithms which can smooth the previous gene prediction drawbacks. Still this requires a proper use of sophisticated classification methods and careful selection of potential DNA sequence features that could best discriminate between coding and noncoding sequences.

2 Materials and Methods

In Figure 1 we depict the overall architecture devised for the comparison of the classifiers. It follows the classical steps of data prepreocessing, learning and test. First, coding and non-coding sequences are extracted for the identification of potential sequence features, and next classification models are built for further prediction analysis (Figure 1-A). Once new sequences are retrieved it is possible to classify them in accordance with the classification models, and thus, an appreciation regarding whether it is a coding sequence or not can be done(Figure 1-B).

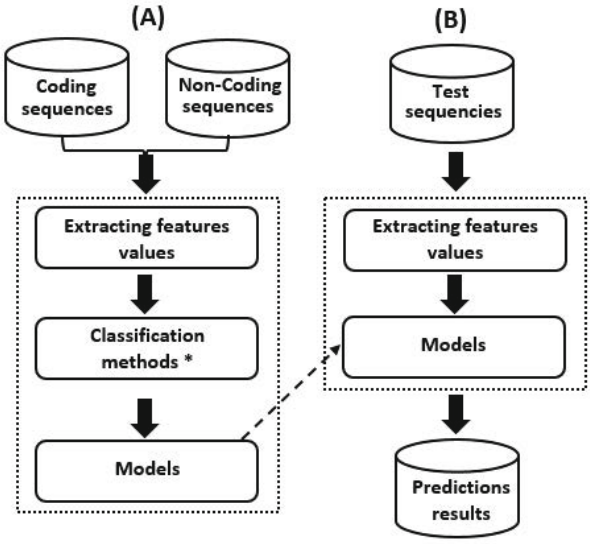


Fig. 1. The overall architecture devised for the comparison of the classification methods

2.1 Classification Methods

We have selected four classification strategies for the comparison study. These methods employ distinct learning strategies, and ideally, each one has a particular manner to generalize the search space. The gene prediction problem is simply a binary classification or *concept learning* (positive class: coding sequence and negative class: no coding sequence). This comparison takes into account distinct supervised learning strategies: one lazy learner (KNN: K-Nearest Neighbors), two eager-learner (SVM: Support Vector Machines and ANN: Artificial Neural Networks) and one ensemble learner (RF: Random Forest). Next, we briefly describes each one of these strategies.

Random forest is a well-known ensemble approach for classification tasks proposed by Breiman [3]. Its basis comes from the combination of tree-structured classifiers with the randomness and robustness provided by bagging and random feature selection.

Nearest-neighbor classifiers are based on learning by analogy, by comparing a given test instance with training instances that are similar to it [4].

A **neural network** is a set of connected input/output units in which each connection has a weight associated with it. During the learning stage, the network learns by adjusting the weights with aims to predict the correct class label of the input instances. *Backpropagation* is the most popular ANN algorithm and it performs learning on a *multilayer feed-forward* neural network [4].

Support Vector Machines uses a linear model to implement nonlinear class boundaries. SVM transform the input using a nonlinear mapping, thus, turning the instance space into a new space [5].

2.2 Feature Engineering

Feature engineering is at the core of classification strategies and it is a crucial step on prediction modeling. Essentially, two different types of information are currently used to try to find genes in a genomic sequence: i) *content sensors* are the characteristic patterns of protein coding sequences; and ii) *signals sensors* are the features of protein coding sequences based on functional characteristics of the gene structures. In this work we use only content sensors.

Table 1. Content sensors features used [x] by gene prediction tools in metagenomics

	GC Content	Length	Codon usage	Dicodon usage	Translation initiation site	Amino acid usage
Orphelia	x	x	x	x	x	
MetaGUN		x	x		x	
MGC	x	x	x	x	x	x
MetaGene	x		x	x		
FragGeneScan			x			

GC-Content. It is the percentage of guanine and cytosine bases in all bases of a sequence. It has been used extensively by several gene prediction tools. This utilization is mainly due to the fact that coding regions present, on average, a higher GC content than on non coding sequences [6]. Differently from previous studies (see Table 1), we calculated the total level of GC content, and the content at the first, second and third monocodon positions with the aim to evaluate their impact in the gene prediction task. In this way, four features are derived from the GC content.

Length. Another feature for discrimination between coding and non-coding sequence is its length. The intergenic regions are usually smaller than coding regions[7].

Codon Usage. Perhaps the most important features for the discrimination between coding and non-coding sequences can be calculated from codon usage [8], in particular the frequencies of 4^3 monocodons. These frequencies represent the occurrences of successive trinucleotides (non-overlapping). For the characterization of monocodon usage, we compute the variance among the 61 monocodons, since gene sequences do not contain stop codons.

2.3 Training Data

The training data is basically DNA sequences having both coding sequences (positive) and intergenic regions (negative) instances. Our approach to compare the four classification methods is based on a learning scheme over eight prokaryotic genomes, namely two *Archaeas* and six *Bacterias*, available in GenBank[2] (Table 2). The choice of these organisms has to do with the experimental genomic data evaluated while testing the predictive models. Thus, either these organisms belong to the same branch of the evolutionary tree or they are associated to Acid Mine Drainage biofilms (Section 2.4).

We have developed an algorithm to properly extract the coding and non-coding regions, on both forward and reverse strands, from these eight "complete" genomes. This algorithm was applied to regions with sequence lengths higher than 59 bp. Sequences less than 60 bp are ignored since they are too short to provide useful information [9]. Those originating from the annotated genes are used as positive instances of coding sequences, whereas others are treated as items of the non-coding class.

2.4 Test Data

The metagenomic data selected for the comparison study is the Acid Mine Drainage (AMD) biofilm [10], freely available at the site of NCBI [3]. This biofilm sequencing project was designed to explore the distribution and diversity of metabolic pathways in acidophilic biofilms. More information regarding the AMD study as well as environmental sequences, metadata and analysis can be obtained at [10].

[2] `http://www.ncbi.nlm.nih.gov/news/10-22-2013-genbank-release198`
[3] `http://www.ncbi.nlm.nih.gov/books/NBK6860/`

Table 2. The prokaryotic genomes used as reference for the training data. The "*" symbol highlights the two *Archaeas*.

Species	GenBank Acc.
Thermoplasma acidophilum *	NC_002578
Thermoplasma volcanium *	NC_002689
Acidimicrobium ferrooxidans	NC_013124
Acidithiobacillus caldus	NC_015850
Acidithiobacillus ferrooxidans	NC_011206
Acidithiobacillus ferrivorans	NC_015942
Candidatus Nitrospira defluvii	NC_014355
Thermodesulfovibrio orangestonii	NC_011296

We have selected prokaryotic genomes associated to the same species found in Tyson[10]. Thus, five genomes (2 *Archaeas* and 3 *Bacterias*) were extracted from GenBank to create the test data (Table 3).

Table 3. The prokaryotic genomes used as reference for the test data. The "*" symbol highlight the two *Archaeas*.

Species	GenBank Acc.
FA: Ferroplasma acidarmanus *	NC_021592
TA: Thermoplasmatales archaeon BRNA *	NC_020892
LFI: Leptospirillum ferriphilum	NC_018649
LFO: Leptospirillum ferrooxidans	NC_017094
SA: Sulfobacillus acidophilus	NC_015757

2.5 Measures of Prediction Performance

The classifiers will be evaluated through the evaluation of classical prediction performance measures, namely, accuracy (ACC) and Kappa. Kappa measures how closely the instances labeled by the classifiers matched the data labeled as *ground truth*, controling for the ACC of a random classifier as measured by the expected accuracy. Thus, the kappa for one classifier is properly comparable to others kappa's classifiers for the same classification task.

3 Results and Discussion

3.1 Performance of the Classifiers

The prediction modeling and evaluation was carried out with the *caret R* package [11]. We use the built-in *tune*() function for resampling and tuning to optimize all classifiers parameters. The best values were as follows: i) RF (mtry 4), KNN (k=5), ANN (size=5 and decay=0.1), SVML (C=0.5). The performance measures were calculated from the average performance of three resampling repetition within a 10-fold cross validation scheme (Table 4).

Table 4. The average performance of the classifiers. The gray cells highlight the best performance achieved by the RF classifier.

	ACC	KAPPA
RF model	**0.94**	**0.87**
KNN model	0.87	0.70
ANN model	0.91	0.80
SVML model	0.88	0.74

3.2 Comparison of Classifiers Using Independent Test Data

As we expected the ensemble learning classifier employed by RF has achieved the best performance among all classifiers (see Table 5). Ensemble learning algorithms are less likely to *overfitting* when dealing with imbalanced data. The SVM has an overall performance similar to KNN (base classifier), and this is partially due to the generalization carried by a linear SVM. Probably a radial SVM model would be able to generalize better the search space. On the other hand, the other eager learner, ANN, presents competitive results. As an example, ANN outperforms RF for the LFI specie (Kappa=0.9097).

Table 5. The comparison performance of classifiers in accordance to the ACC and Kappa measures. The highlighted cells show the best results.

	ACC				Kappa			
Species	RF	ANN	KNN	SVML	RF	ANN	KNN	SVML
FA	0.9173	0.8702	0.8302	0.785	**0.8275**	0.7298	0.6182	0.5317
LFI	0.9156	0.9097	0.8854	0.8835	**0.8256**	**0.9097**	0.7599	0.7565
LFO	0.9263	0.9143	0.8888	0.8767	**0.8472**	0.8213	0.7666	0.741
SA	0.9383	0.9235	0.8913	0.8947	**0.8741**	0.8434	0.7746	0.7834
TA	0.957	0.9175	0.8875	0.9175	**0.9089**	0.8243	0.7577	0.737

3.3 Random Forest Classifier Evaluation

In Figure 2 we show the importance of each of the 6 variables used by RF. The size of the sequence was selected as the most important attribute, maybe due to the fact that the coding regions are in most cases higher than non-coding. Among the GC content measures applied, the concentration of GC in second position of codons showed a very significant importance as compared to the others. Finally, the variance of the codon usage did not show a higher degree of relevance as expected.

Based on the rank of the most important features illustrated in Figure 2, we have built three other RF models as follows: RF5) does not take into account the GC content feature; RF4) does not use features related to the GC content and the GC content in the third position; and RF3) does not take into account the features GC content and GC content in first and third position. Finally, RFComp uses the complete set of features. From Figure 3 we may observe that the features derived from the GC content plays an interesting role the in models' generalization.

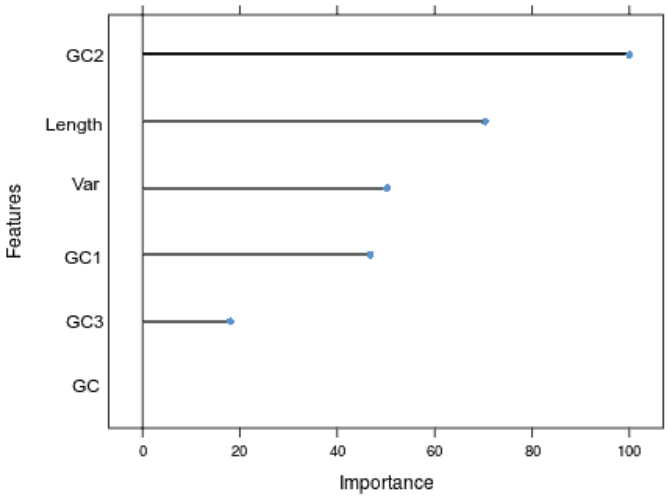


Fig. 2. Variable importance plot of the RF model

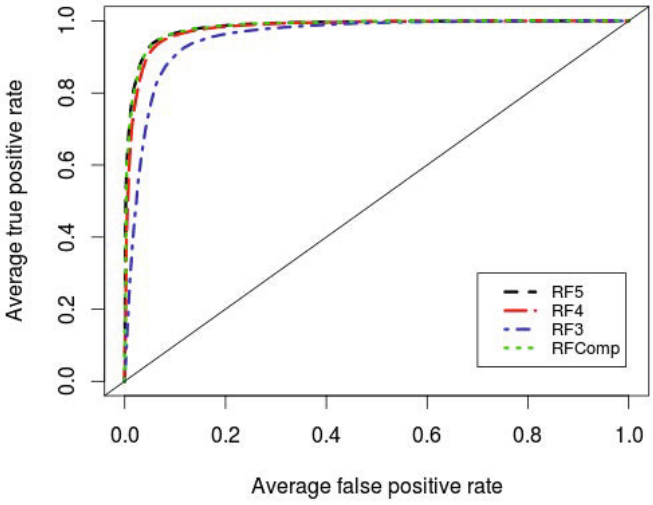


Fig. 3. The ROC curve for distinct RFn models. RFcomp uses the complete set of six features, the other ones uses (n) features on each model. Thus, RF3 uses only the Top-3 features from the Figure 2.

4 Conclusions

Gene prediction is a well-known computational challenge in both genome and metagenome analysis. In this work we presented an empirical comparison of several well-known classification methods applied to gene discovery in experimental metagenomic data. Though the performance of the four base classifiers was good, the ensemble-based strategy *Random Forest* has achieved the overall best result. We plan to develop a new gene prediction *pipeline* having its basis on Random Forest. To the extent of our knowledge there is no reference of a metagenome gene prediction strategy based on a RF classifier.

Author's Contributions

FG and RA performed the analysis and developped the pipeline. RA and CC supervised the study. FG, RA, CC and LT wrote the manuscript.

Acknowledgements. This work is partially supported by CNPq under the BIOFLOWS project [475620/2012-7]. FG has also master scholarship by Capes.

References

1. Wooley, J.C., Godzik, A., Friedberg, I.: A primer on metagenomics. PLoS Computational Biology 6(2), e1000667 (2010)
2. Kunin, V., Copeland, A., Lapidus, A., Mavromatis, K., Hugenholtz, P.: A bioinformatician's guide to metagenomics. Microbiology and Molecular Biology Reviews 72(4), 557–578 (2008)
3. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
4. Han, J., Kamber, M., Pei, J.: Data mining: concepts and techniques. Morgan kaufmann (2012)
5. Faceli, K.: Inteligência artificial: uma abordagem de aprendizado de máquina. Grupo Gen-LTC (2011)
6. Fickett, J.W.: Recognition of protein coding regions in dna sequences. Nucleic Acids Research 10(17), 5303–5318 (1982)
7. Mathé, C., Sagot, M.F., Schiex, T., Rouzé, P.: Current methods of gene prediction, their strengths and weaknesses. Nucleic Acids Research 30(19), 4103–4117 (2002)
8. Hoff, K.J., Tech, M., Lingner, T., Daniel, R., Morgenstern, B., Meinicke, P.: Gene prediction in metagenomic fragments: a large scale machine learning approach. BMC Bioinformatics 9(1), 217 (2008)
9. Liu, Y., Guo, J., Hu, G., Zhu, H.: Gene prediction in metagenomic fragments based on the svm algorithm. BMC Bioinformatics 14(suppl. 5), S12 (2013)
10. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., Banfield, J.F.: Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature 428(6978), 37–43 (2004)
11. Kuhn, M.: The caret package homepage (2010), `http://caret.r-forge.r-project.org`

FUNN-MG: A Metagenomic Systems Biology Computational Framework

Leandro Corrêa[1], Ronnie Alves[1,2,4], Fabiana Goés[1], Cristian Chaparro, and Lucinéia Thom[3]

[1] PPGCC, Universidade Federal do Pará, Belém, Brazil
hscleandro@gmail.com
[2] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, UMR 5506, Université Montpellier 2, Centre National de la Recherche Scientifique, Montpellier, France
alvesrco@gmail.com
[3] PPGC, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil
lucineia@inf.ufrgs.br
[4] Institut de Biologie Computationnelle, Montpellier, France

Abstract. Microorganisms abound everywhere. Though we know they play key roles in several ecosystems, too little is known about how these complex communities work. To act as a community they must interact with each other in order to achieve such *community stability* in which proper functions allows the microbial community to adapt in complex environment conditions. Thus, to effectively understand microbial genetic networks one needs to explore them by means of a systems biology approach. The proposed approach extends the metagenomic gene-centric view by taking into account the set of genes present in a metagenome and also the complex links of interactions among these genes and by treating the microbiome as a single biological system. In this paper, we present the FUNN-MG computational framework to explore functional modules in microbial genetic networks.

Keywords: systems biology, gene and pathway enrichment analysis, graph representation, graph visualization, metagenomics.

1 Introduction

Microorganisms abound in every part of the biosphere including soil, hot springs, on the ocean floor, high in the atmosphere, deep inside rocks within the Earth's crust and in human tissues. They are extremely adaptable to conditions where no one else could be able to survive.

Their adaptability is mainly due to the fact that they live in complex communities. Interactions inside the microbial networks plays essential functions for the maintenance and survival of the community. Unfortunately, too little is known about microbial interactions.

In this sense, the metagenomic analysis conducts studies from of material extracted directly from the environment has been developed in order to extract

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 25–32, 2014.

knowledge from this interaction. The metagenomic sequence process poses challenges that could be handled by the utilization of Machine Learning (ML) techniques. In fact, ML has been applied succesfully in several genomics problems. In the context of functional analysis it can provide new ways to explore graphs by using robust statistics, dealing with uncertainty in the data and boosting the search for "hot spots" in large microbial genetic networks.

In this work we propose a computational framework to evaluate functional modules in microbial genetic networks. A weighted graph is built with its basis on the genes and pathways properly induced from the relative abundance of the metabolic pathways enriched by the associated metagenomic data. Aditionaly, non-supervised ML is applied to enumerate network modules (clusters) of microbial genes presenting strong evidence of both interaction and functional enrichment.

The main contribution of the proposed strategy are:

- A systems biology approach for functional enrichment analysis of metagenomes;
- A visual analytics system to explore interactively the enriched metabolic pathways in microbial genetic networks;
- the FUNN-MG computational framework for the identification of network modules having strong functional enrichment in experimental metagenomic data.

2 Metagenomic Pathway-Centric Network Analysis

Metagenomic data analysis is a complex analytical tasks in both biological and computational senses. To help visualize the processes involved between species found in metagenomic samples and decrease the complexity of the analyzes, in this work we propose the FUNN-MG computational framework (Figure 1) which provides a functional and visual analytic system for the identification and exploration of the *key* functions of a microbial community. The FUNN-MG is acronym for FunctioNal Network Analysis of Metagenomics Data.

The proposed strategy has four main tasks (the rounded rectangles in Figure 1) that must be executed sequentially: i) identification of the metabolic pathways, ii) statistical evaluation of the enriched pathways, iii) detection of strong modules (clusters) and iv) visualization of the microbial gene-pathway network. The first three steps are related to the ML part of the strategy while the remaining step deals with the visual analytics of the graph patterns extracted in the previous steps. Next section we discuss each one of these steps, leaving one particular section to the visualization strategy.

3 Materials and Methods

3.1 The Metagenomic Experimental Data

The metagenomic data selected for our experimental study is the Acid Mine Drainage (AMD) biofilm [1], freely available at the site of NCBI[1]. This biofilm

[1] `http://www.ncbi.nlm.nih.gov/books/NBK6860/`

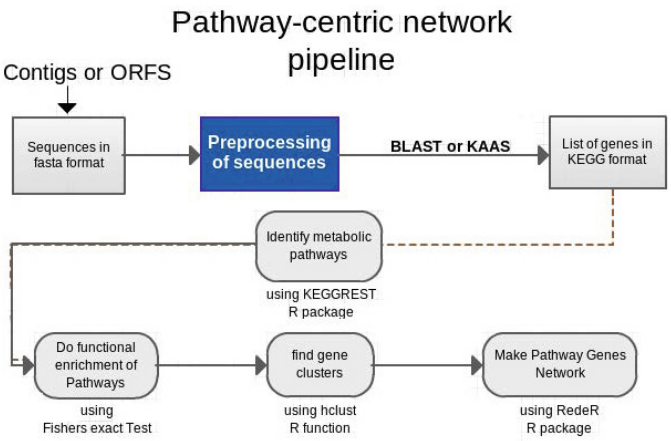


Fig. 1. Metagenomic pathway-centric network approach for functional and visual analytics of microbial communities

sequencing project was designed to explore the distribution and diversity of metabolic pathways in acidophilic biofilms. Acidophilic biofilms are self-sustaining communities that grow in the deep subsurface and receive no significant inputs of fixed carbon or nitrogen from external sources. More information regarding the AMD study as well as environmental sequences, metadata and analysis can be obtained at [2].

3.2 Preprocessing of the Metagenomic Sequences

We have used the KAAS tool [3] for the identification of 477 microbial genes divided into six different species (Figure 2). This identification was based on the nucleotide similarity of the groups of orthologous genes found in the KEGG database [4].

The search for microbial genes was carried out in several steps. First, identify the species corresponding in the KEGG database [2] (dendogram Figure 2). Thereafter, KAAS tool was employed and some analysis were performed sequentially in four steps: i) finding groups of orthologous genes, ii) identification of associated species in KEGG, iii) obtaining functional annotation in KEGG database, iv) Eliminating duplicated genes. All the steps above were executed for all reference species in the AMD sample, taking into account its associated target species in the KEGG database.

3.3 Identifying Metabolic Pathways

The "*KEGGREST*" *R* package [5] was applied using as reference the list of 477 genes identified, highlighting 95 pathways for the AMD metagenome. Though at this step we cannot assume any strong evidence of functional enrichment regarding to the genes identified.

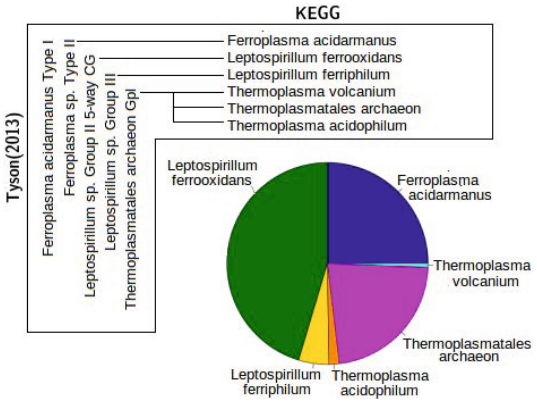


Fig. 2. The dendrogram on the top highlights the association between species in the AMD metagenome and its target species in the KEGG database. In the bottom, a pie chart of the distribution of the 477 genes identified.

3.4 Functional Enrichment Analysis

To calculate the functional enrichment of metabolic pathways we apply Fisher's exact test[2]. The main challenge in evaluating the enrichment of a metabolic pathways is the calculation of the probability of finding species covered on each pathway across samples, given that, eventually, only a selected group of species will have an associated pathway. This is also due to the fact that species play distinct roles in the microbial community.

Table 1. The contigency table of the *Glutathione metabolism* pathway which is required for the calculation of the enrichment score

	Gene associated with a pathway	Gene not associated with a pathway	Total gene
Sample	**a** **(6)**	**b** **(364)**	a+b (370)
Population	**c** **(15)**	**d** **(2768)**	c+d (2783)
Total in KEGG	a+c (21)	b+d (3132)	n (3153)

As an example, the metabolic pathway *Glutathione metabolism* is annotated for five out of six species identified in the samples: *Ferroplasma acidarmanus*, *Leptospirillum ferrooxidans*, *Leptospirillum ferriphilum*, *Thermoplasma acidophilum* e *Thermoplasma volcanium*. So, KEGGREST will only take into account these five species for the enrichment score (Fisher's exact test).

[2] `http://en.wikipedia.org/wiki/Fishers_exact_test`

In Table 1 we present the contigency table required to calculate the enrichment of the *Glutathione metabolism* pathway with respect to the microbial genes found in the samples and its corresponding annotations in KEGG. Having this table, we use the *phyper* function in the "*stats*" R package for the enrichment score, followed by a test of significance using the *"Firsher's exact test for count data"* R package. Finally, we obtained an enrichment score of 0.0077 (p-value = 0.0292) for the the *Glutathionemetabolism* pathway.

After completing the functional analysis for the 95 metabolic pathways, we obtained a list with only 11 enriched pathways (p-value ≤ 0.05) corresponding to 329 genes.

3.5 Finding Gene Clusters

Once identified the set of significant pathways in the sample, we explore functional modules presenting strong gene interactions by the utilization of a bipartite graph structure $MGP = (G, P, E)$. We called this bipartite graph *Microbial Gene Pathway* (Figure 3. a). MGP vertices are divided into two disjoint sets (G)enes and (P)athways, such that every edge (E) connects a vertice in (G) to one in (P). The enrichment score is annotated in each vertice (P).

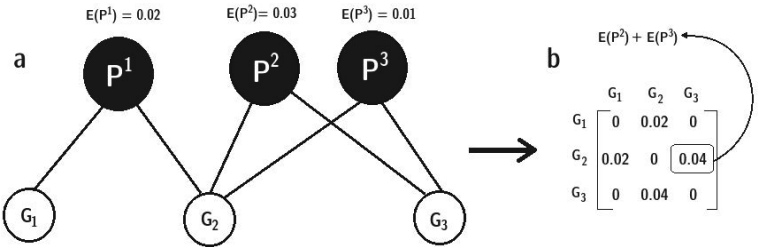


Fig. 3. a) The MGP bipartite graph with (G)enes and (P)athways . b) the associated community matrix with the gene-to-gene interaction augmented with the enrichment score.

The structural graph clustering uses a community matrix (Figure 3.b) based on the genes and its enriched pathways represented in MGP. The community matrix observes three main aspects regarding gene-to-gene interactions: the existence of one or more metabolic pathways shared by the genes, the amount of metabolic pathways in which genes play, and the enrichment score associated to each metabolic pathway.

The MGP bi-partite graph is an interesting computational structure for both the application of ML techniques and interactive visualization of the microbial genetic network [6]. The *community patterns* are obtained directly through the utilization of a hierarchical clustering (*hclust()* R function) technique over the community matrix.

Hierarchical complete clustering solution requires an euclidean distance matrix that can be built directly through the community matrix. Given that we were looking for compact clusters we decided to use the cutting result obtained with the Dynamic hybrid approach through the Dynamic Tree Cut library for R[3]. Thus, 9 clusters and 10 nested subclusters were enumerated. All clusters have the prefix “NT" followed by a sequential number.

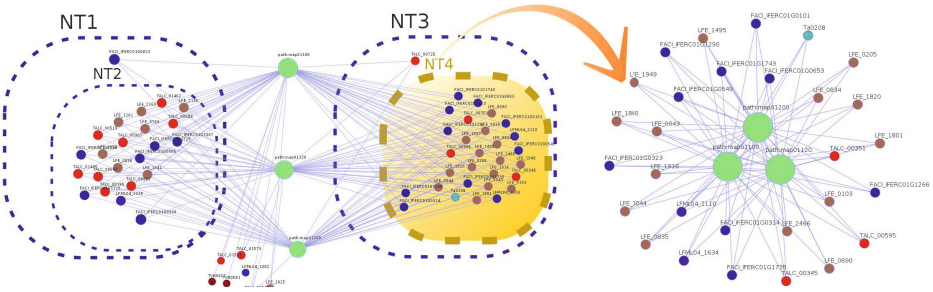


Fig. 4. The representation of the community patterns as clusters (NT1, NT3) and subclusters (NT2, NT4). At the right the expanded subnetwork corresponding to elements clustered in NT4.

In summary, 308 genes were clustered, corresponding to 96.61% of the enriched pathways related to AMD biofilm. These clusters enclose on average 30 genes, having 6 genes in the most compact cluster and 128 in the largest one. The Figure 4 shows the interaction between two clusters (NT1 and NT3) and subgroups (NT2 and NT4) identified.

4 Results and Discussion

4.1 Visual Analytics System

The *MGP* bipartite graph fits properly the graph structure required for visualization through the *RedeR R* package. This network visualization system allows several interactive and graph functions such as: zoom, pan, neighborhood highlighting, search, flows, etc. An example of the structural visualization of the enriched *Microbial Gene Pathway* is presented in Figure 5. The visualization model allows the identification of genes across species and pathways, depicted in distinct colors. It is also possible to explore the degree of connectivity by inspecting the size of the vertices; key players are identified by neighborhood highlighting while clicking on a particular node in the graph network.

[3] `http://labs.genetics.ucla.edu/horvath/CoexpressionNetwork/BranchCutting/`

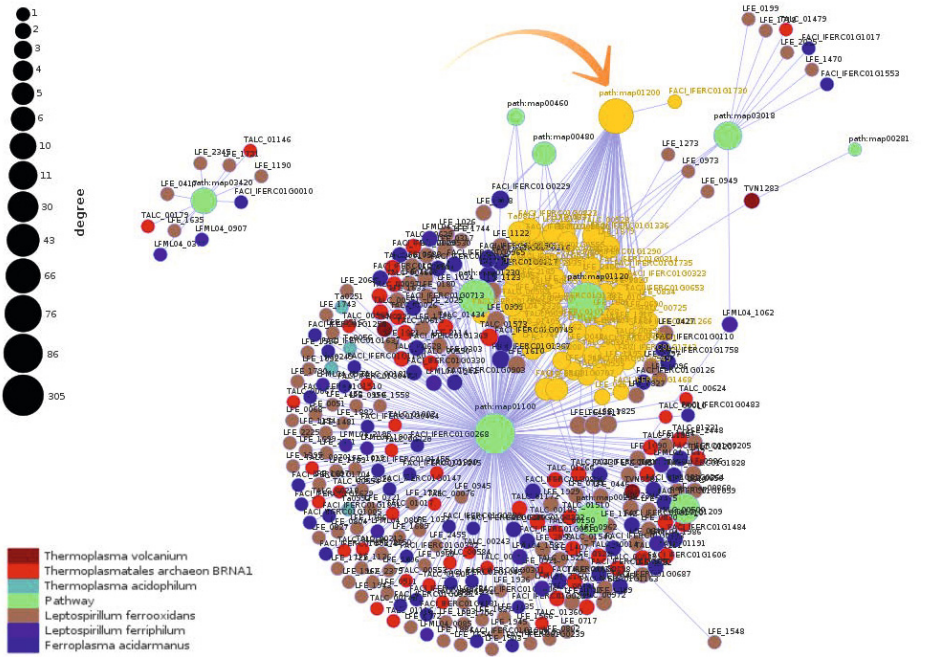


Fig. 5. The enriched Microbial Gene Pathway Network. At the bottom left the legend of the species and associated pathways are represented. Nodes (circles) are related to either species genes or pathways. At the upper left the degree connectivity scale of all nodes. The nodes in highlighting (yellow) are all genes associated to the *Carbon metabolism* pathway (direct orange arrow).

5 Conclusions

The enrichment analysis of microbial genetic networks is an open computational challenge. The enumeration of all gene-to-gene interaction of a microbial community is not practical, therefore pathway-centric analysis sound a promising strategy to smooth this combinatorial aspect of this problem.

We have presented the FUNN-MG computational framework for functional and visual analytics of metagenomes by focusing the enrichment analysis on the identification of community patterns. This strategy has it basis on non-supervised machine learning over a bipartite graph properly built to evaluate the enriched microbial gene pathways. Community patterns are carefully extracted by evaluating strong interactions among gene species sharing the most significant enriched pathways. Once all the topological network aspects are understood for a particular metagenome, we envisage the possibility of using such *community profiles* for metagenome comparison as well as classification of unknown microbial genetic network.

Author's Contributions

LC and RA performed the analysis and developped the pipeline. RA and CC supervised the study. LC, RA, CC and LT wrote the manuscript.

Acknowledgements. This work is partially supported by the Brazilian National Research Council (CNPq – *Universal calls*) under the BIOFLOWS project [475620/2012-7].

References

1. NCBI: Metagenomics: Sequences from the environment [internet]. Sequences from the Environment, Tyson (2013)
2. Tyson, G.W., Chapman, J., Hugenholtz, P., Allen, E.E., Ram, R.J., Richardson, P.M., Solovyev, V.V., Rubin, E.M., Rokhsar, D.S., Banfield, J.F.: Community structure and metabolism through reconstruction of microbial genomes from the environment. Nature 428(6978), 37–43 (2004)
3. Moriya, Y., Itoh, M., Okuda, S., Yoshizawa, A.C., Kanehisa, M.: KAAS: an automatic genome annotation and pathway reconstruction server. Nucleic acids research 35(Web Server issue), W182–W185 (2007)
4. Kanehisa, M., Goto, S., Sato, Y., Furumichi, M., Tanabe, M.: KEGG for integration and interpretation of large-scale molecular data sets. Nucleic Acids Research 40(Database issue), D109–D114 (2012)
5. Tenenbaum, D.: KEGGREST: Client-side REST access to KEGG. R package version 1.0.1
6. Goh, K.I., Cusick, M.E., Valle, D., Childs, B., Vidal, M., Barabási, A.L.: The human disease network. Proceedings of the National Academy of Sciences 104(21), 8685–8690 (2007)

FluxMED: An Adaptable and Extensible Electronic Health Record System

Alessandra C. Faria-Campos[1,2], Lucas Hanke[1], Paulo Henrique Batista[1], Vinícius Garcia[1], and Sérgio Campos[1]

[1] Universalization Access Laboratory, LUAR, Department of Computer Science, Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Pampulha, 30123-970, Belo Horizonte, MG, Brazil
[2] INMETRO, Instituto Nacional de Metrologia, Qualidade e Tecnologia
Av. Nossa Sra. das Graças, 50, Xerém, 25250-020, Duque de Caxias, RJ, Brasil
{alessa,scampos}@dcc.ufmg.br,
{hankelucas,batista.phs,vingarcia00}@gmail.com

Abstract. The amount of data generated by medical and laboratory services grows each day. The number of patients is increasing, modern examination methods generate large amounts of data and the growing specialization of the medical profession makes the problem of storing and managing this data very complex. Computer applications known as Laboratory Information Management Systems (LIMS) have been proposed as tools to address this issue. In this work we propose the FluxMED system, a fully customizable EHR system with an easy to adapt interface for data collection and retrieval. FluxMED can easily be customized to manage different types of medical data. The customization for a new disease can be done in a few hours with the help of a specialist. We have used FluxMED to manage data from patients of three complex diseases, neuromyelitis óptica, paracoccidioidomycosis and adreno-leukodistrofy. These diseases have very different symptoms, different exams are performed to come to a diagnostic and have different treatments. However, FluxMED is able to manage these data in a highly specialized manner without any modifications to its code.

Keywords: Electronic Health Record, Laboratory Information Management Systems, Workflow.

1 Introduction

The storage and management of large amounts of laboratory and medical data has been frequently discussed as the need for software tools that support the entire life cycle of data (collection, storage, analysis, reporting and archiving) is increasing on a daily basis. One of the solutions proposed to address this issue is the use of Laboratory Information Management Systems (LIMS). LIMS are computational tools developed to integrate and manage laboratory data that give emphasis to quality assurance and aim to generate results in a consistent and reliable way [6]. Several LIMS are available nowadays as academic, proprietary

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 33–40, 2014.

and open source applications. Some examples include SQL LIMS [1], LabSoft LIMS [3], LabWare LIMS [14] (proprietary applications), FreeLIMS [4] an open source application developed by Labmatica and the academic systems developed by Hendrick [5], Quo [10], Tharayil [15] and Sanchez [12]. Melo [9] has proposed a new system — SIGLa — a workflow based LIMS designed to allow it to adapt its activities and processes to various types of laboratories. A workflow can be defined as a sequence of steps and tasks executed according to a set of rules and procedures in order to complete a process.

The need for specific LIMS is particularly important for medical laboratories and facilities since for medical applications, existing systems are frequently focused on maintaining the doctors schedule and keeping general annotations on the patients conditions. As a consequence, the data stored cannot be used for a detailed analysis and cannot be easily integrated with other systems.

Existing systems tend to fall in one of two categories: They can be too rigid in the types of data that can be stored, limiting severely the symptoms, exams, diagnostics that can be recorded. As a way of compensating for this problem, the other type of system is too generic, allowing the doctor to enter free text describing the patients consultation. Data entered in this way is very difficult to analyze since data from different consultations often cannot be compared [2],[13].

FluxMED takes a different approach, defining the types of data entered in the workflow. These can be changed easily, incorporating new knowledge without changes to the system. It can be used in very flexible ways, for example, if different doctors follow different diagnostic strategies, that is, ask different questions, and request different exams, the workflow can incorporate both methods, and let the doctor choose which one to use.

Data entered in this way is structured to make it easy to analyze it later. Data is not entered in free text format, but in formats that have fixed types and requirements, which simplifies posterior analysis.

We have used FluxMED to develop EHR systems for three different diseases that are complex, difficult to diagnose and to treat. But because they are not common diseases, EHR systems aimed at them are non existent or very difficult to access. FluxMED has been able to model data from patients of neuromyelitis óptica, paracoccidioidomycosis and adrenoleukodistrofy and enable doctors to use the system to treat their patients. Data from these datasets will later be used on data analysis systems to identify patterns and conditions that can help treating the patients and improving their life.

An important aspect of the FluxMED system is that creating a workflow for a new disease takes only a few hours with the help of a specialist. There is no need to change the system in any way. With some training the doctors can themselves specify the workflow and create the EHR system. Moreover, new systems can be integrated with existing ones, so one EHR system can serve several specialities, making it simpler to maintain the data, train users and extend the system. FluxMED allows users to compare data between different but related diseases in search of common aspects that can be considered for a treatment, but which would have been very difficult to identify if no integrated system is available.

FluxMED is based on the SIGLa system [9]. It has an easy to use Web interface. It has several security features that prevent unauthorized access, including different permission sets for users to access different types of data depending on their authorization. The system can be accessed at the address below with login/password: `guest/gu3st`. All data in the web site and in this paper is fictitious and do not represent real patients. `http://syrah.luar.dcc.ufmg.br/FluxMED`

2 FluxMed Development

The FluxMED system has been constructed using Java and uses MySQL as database server and Apache Tomcat as Web server. Workflow files are uploaded in the system through the interface and the activities created in the workflow construction are interpreted as links by the system. The workflows are constructed in the XPDL format using the application Together Workflow Editor (TWE) [16]. In FluxMED the EHR systems are defined as sequences of activities. An activity represents events such as a consultation, an exam or test performed, or a diagnostic determined. Specific information, such as names, dates, values for specific exams or doctor analysis are represented as attributes in the workflow definition. Therefore, the user can define the characteristics of the attributes of each activity, such as its types, their range of values, its formats or even define auto-calculated attributes generated from other attributes.

Several types of attributes contribute to the generality of the FluxMED implemented EHRs. Traditional types of attributes are supported, such as integer or real numbers, which can have specified ranges, or strings containing text. Other types include the register attribute, which ensures that the values entered are already present in a separate table. For example, the name of the doctor may be specified to contain only values that have been already registered in the system, avoiding problems with misspelling names. All of these types of attributes and their functionalities can be specified in the workflow, and can be added or modified by the user without changing system code.

The FluxMED System has an easy to use interface where activities that have already been executed are represented by icons in a different color from those for activities available to be executed (Figure 1 shows part of the screen). The system guides the user through the entire process, informing which activities are available for execution. Adittionally, this execution can also be done automatically through the upload of a file containing the attributes values. The workflow is entirely modeled and built on TWE in the XPDL file format and contains all the information regarding the entire process. As a consequence, to change the type of data being stored, only the workflow needs to be modified.

3 EHR Systems Developed

To illustrate the usefulness of FluxMED, we have modelled and implemented EHR systems for three diseases, neuromyelitis óptica (NMO), paracoccidioidomycosis and adrenoleukodistrofy. We will detail the NMO system, and briefly mention the others, but the details of the modeling process are similar. All systems

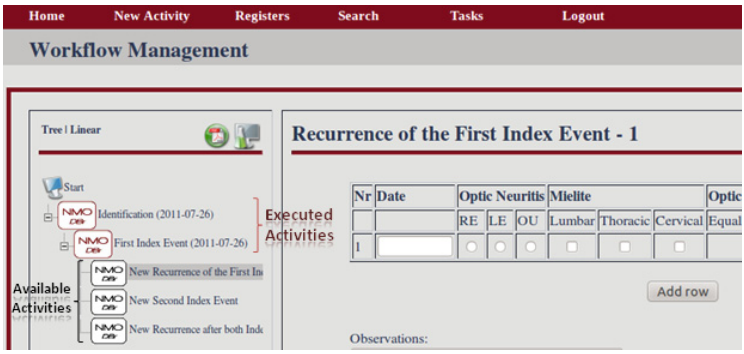


Fig. 1. NMO Interface where the already executed and available activities are shown in different colors

have been specified by a specialist in the disease, and the workflow development has been done by a informatics student working with the specialist. Tipically development takes a few hours spread over a number of sessions which interactively refine the workflow until the system is considered finished. All systems have been tested by doctors which verified that the system can be used for real patient consultation. Two systems, NMO and ALD have not yet been deployed in actual consultations. The PCM system has been used already to store data from several patients, and doctors are now completing the database with previous consultation data to allow the continuation of the treatment in the system and a complete analysis of existing patient data.

3.1 Neuromyelitis Optica—NMO

Neuromyelitis optica is an important central nervous system disease. NMO is an idiopathic inflammatory demyelinating disease of the central nervous system (CNS) most frequently characterized by recurring attacks of optic neuritis and myelitis. It can be distinguished from conventional multiple sclerosis on demographic, clinical, neuroimaging, cerebrospinal fluid and serological grounds [17]. NMO spectrum disorders are highly prevalent among the demyelinating diseases of the CNS in Southeastern Brazil and a significant amount of data from NMO patients is generated from medical care. The collection and analysis of these data will result in a better characterization of the disease and therefore, there is a strong need for computational tools and databases to collect, store, manage and retrieve NMO data in order to help improve research and medical care. This work has been done in cooperation with Dr. Marco Lana, from the Center for Investigation of Multiple Sclerosis at UFMG (CIEM).

A complete workflow for storage and management of data from NMO patients has been created to be used in the FluxMED System, along with a database for data storage [8]. The workflow consists of 16 activities (Figure 2) and each of those has an individual screen in the system. The first activity is the *Identification* which requires general information from the patient, such as name, age

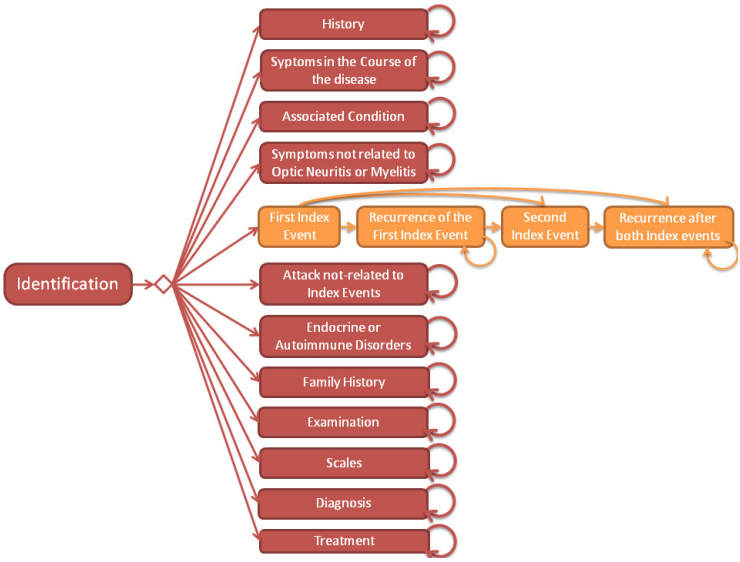


Fig. 2. NMO workflow definition with the 16 steps of the process

and level of education. Once the patient's identification is completed, several activities are available for execution (Figure 1). Each activity has its own set of attributes and the formats for the data are defined during the workflow definition. These activities represent all the steps used in medical evaluation of NMO patients.

3.2 Paracoccidioidomycosis—PCM

Paracoccidioidomicosis (PCM) is a typical Brazilian disease, caused by the yeast Paracoccidioides brasiliensis. The most common form of infection is through inhalation of spores. The spores cause infection of the lung epithelium and can spread to other organs. The disease mainly affects farm workers who are exposed to contaminated soil during labor [11]. This disease represents an important Public Health issue, due to its high incapacitating potential and the amount of premature deaths it causes if untreated. The analysis and management of PCM related data presents several challenges. One of the challenges is related to data acquisition during patient evaluation and diagnosis. The Center of Training and Reference on Infeccious-Parasitary Diseases from the Federal University of Minas Gerais (CTR-DIP-UFMG), coordinated by Dr. Enio Pietra, has developed a protocol for clinical analysis of PCM patients. This protocol includes a large number of clinical variables that are assessed in each medical examination, including x-ray and serology tests, which are also used in tracking the disease progression. We have modelled this protocol in a workflow and used it to create an EHR system in FluxMED. This system is currently in use at UFMG, where it is helping manage patient's data and assisting in defining treatment duration and other conditions (Figure 3).

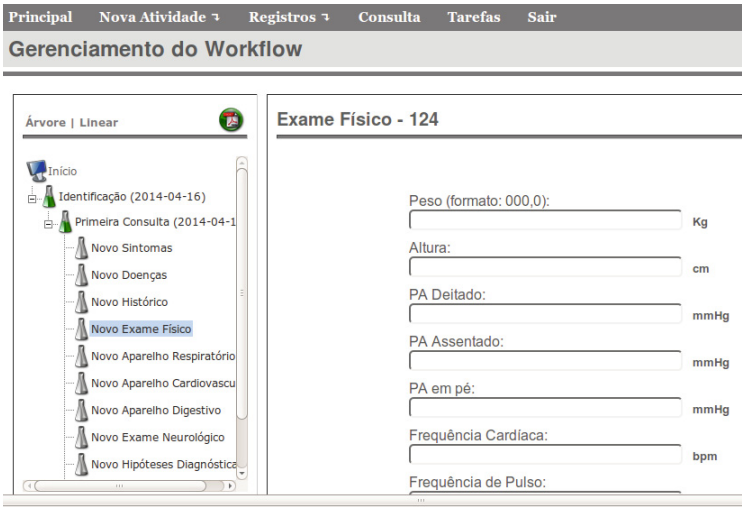


Fig. 3. FluxMed Interface showing the electronic forms for the PCM EHR system

3.3 Adrenoleukodistrofy—ALD

The Inborn Errors of Metabolism Laboratory at UFMG, coordinated by Dr. Eugenia Valadares, works with patients from the public health system in Brasil, identifying and treating rare genetic diseases that are difficult to diagnose but often have treatments available if diagnosed early. These diseases can seriously affect the people afflicted, and can manifest at any age. Each disease is in isolation rare, but are expressive when different diseases are accounted for. With an early diagnosis, frequently treatments are effective, not only helping patients to have a better life, but also lowering treatment costs of these patients throughout their lives.

In our work with the UFMG laboratory we plan to model several of these diseases. Currently we have modelled adrenoleukodistrofy, or ALD [7]. ALD is a disorder of peroxisomal fatty acid beta oxidation which results in the accumulation of very-long chain fatty acids in tissues throughout the body. The most severely affected tissues are the myelin in the central nervous system, the adrenal cortex and the Leydig cells in the tests. Early diagnosis and treatment can reduce or eliminate symptoms during the lifetime of patients, and is essential, but also difficult, due to the fact that the disease is rare and complex to diagnose. We have modelled the ALD protocol and created an EHR system to manage this data. We are currently registering in the system data from patients from the laboratory, which number to about 400 per year.

4 Discussion

The use of workflows to develop an adaptable LIMS such as the FluxMED has resulted in the construction of a very flexible tool. All changes on the activities flow affect only the workflow, since it belongs to an independent layer on the

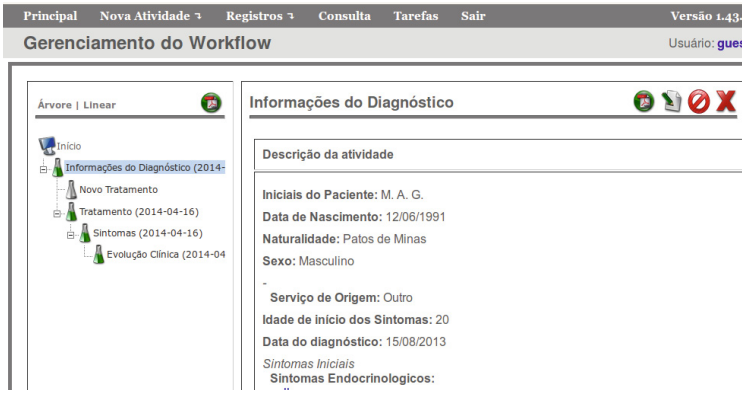


Fig. 4. FluxMed Interface showing the electronic forms for the ALD EHR system

system. Therefore changes to adapt the system to the needs of a new laboratory or facility are easy to implement and do not require any programming skills from the user. In this work, workflows have been developed by computer science students working with medical specialists. Medical staff, however, can easily be trained to create new workflows or adapt existing ones, creating a dynamic environment where management of new diseases and conditions can be added to the system by its users.

One of the key advantages of the FluxMED system is that it integrates all EHR systems. Each new workflow representing a new disease or specialty is added to the same EHR system and integrates all data in one database. This makes it easier to deploy the EHR system, since all systems will use the same interface, and new ones are added with minimal changes from existing ones.

In addition to this, having all data in one database makes it possible to relate symptoms, treatments and other conditions from different diseases, and to identify common characteristics and key differences between them. Analyzing this data will make it possible not only to generate new scientific knowledge comparing different diseases, but will also enable the system to guide the user depending on the data entered, suggesting diagnostics that can assist the doctors using the system. Notice that this is not possible if data from different diseases are not stored in an integrated database.

The construction of the NMO, PCM and ALD data management systems represent an important contribution to help improve data collection from patients and can assist significantly in their treatments. All of these workflows are highly complex and specialized, building an EHR system for each would not have been practical and would not have been done without the FluxMED system.

5 Conclusions

The advances on medical research and procedures resulted in the generation of a large amount of medical data. The storage and management of this information has become incompatible with the use of paper medical charts and computer

flat files, requiring an automated system. The adoption of more powerful tools to address this issue is an immediate need in many clinics and medical offices, particularly for those working on complex diseases that are difficult to diagnose. In this work, we have developed FluxMED, a workflow based system for that purpose with a modern and user friendly interface designed to assist medical doctors on the diagnosis and monitoring of their patients. The system is flexible and can be modified by the user to model different types of diseases, contributing to make diagnosis faster and more reliable, helping to distinguish complex diseases from other related ilnesses, shortening treatment times and improving patients quality of life. FluxMED is currently being used in three different laboratories or clinics at UFMG, assisting doctors and patients.

References

1. (June 2011), http://www.labware.com/lwweb.nsf, http://www.sqllims.com
2. Asabi, S., Oye, N., Goji, M.: Hospital patient database management system. COMPUSOFT 2(3), 65–72 (2013)
3. Computing Solutions, I. (June 2011), http://www.labsoftlims.com
4. Geeknet, I.: Freelims (September 2011), http://sourceforge.net/projects/freelims/
5. Hendricks, M., Learn, R.W.: A laboratory information management system (lims) for an academic microchip fabrication facility. In: University/Government/Industry Microelectronics Symposium (2003)
6. Hinton, M.D.: Laboratory Information Management Systems: development and implementation for a quality assurance laboratory. Marcel Dekker, New York (1995)
7. Kwon, J.: Neurodegenerative disorders of childhood. In: Kliegman, R.M., Behrman, R.E., Jenson, H.B., Stanton, B.F. (eds.) Nelson Textbook of Pediatrics (2012)
8. Lana-Peixoto, M., Talim, L., Faria-Campos, A., Campos, S., Rocha, C., Hanke, L., Talim, N., Batista, P., Araujo, C., Kleinpaul, R.: Nmo-dbr: the brazilian neuromyelitis optica database system. Arq Neuropsiquiatr 69(4), 687–692 (2011)
9. Melo, A., Faria-Campos, A., DeLaat, D., Keller, R., Abreu, V., Campos, S.: Sigla: an adaptable lims for multiple laboratories. BMC Genomics 11(5), S8 (2010)
10. Quo, B.W.M., Wu, C.F.: Development of a laboratory information system for cancer collaboration projects. In: 27th Annual International Conference of the Engineering in Medicine and Biology Society (2005)
11. Restrepo, A., McEen, J., Castaneda, E.: The habitat of paracoccidioides brasiliensis: how far from solving the riddle? Med. Mycol. 39, 233–241 (2001)
12. Sanchez-Villeda, H., et al.: Development of an integrated laboratory information management system for the maize mapping project. Bioinformatics, 2022–2030 (2003)
13. Sim, K., Chong, S., Tso, C., Nial, M., Chong, A., Abbas, S.: Computerized database management system for breast cancer patients. SpringerPlus 3(268), 1–16 (2014)
14. Solutions, L.L. (June 2011), http://www.labware.com/lwweb.nsf
15. Tharayil, T.K.A., Kalbfleisch, S.M.: Service-oriented laboratory information management system for life sciences research. In: International Conference on Services Computing, pp. 621–627 (2007)
16. Together-Workflow-Editor (2011), http://www.together.at/prod/workflow/twe
17. Wingerchuk, D.M., Hogancamp, W.F., Brien, P.C., Weinshenker, B.G.: The clinical course of neuromyelitis optica (devics syndrome). Neurology (1999)

Influence of Sequence Length in Promoter Prediction Performance*

Sávio G. Carvalho, Renata Guerra-Sá, and Luiz H. de C. Merschmann

Federal University of Ouro Preto (UFOP), Ouro Preto/MG, Brazil
saviogcarvalho@yahoo.com.br,
{rguerra,luizhenrique}@iceb.ufop.br

Abstract. The advent of rapid evolution on sequencing capacity of new genomes has evidenced the need for data analysis automation aiming at speeding up the genomic annotation process and reducing its cost. Given that one important step for functional genomic annotation is the promoter identification, several studies have been taken in order to propose computational approaches to predict promoters. Different classifiers and characteristics of the promoter sequences have been used to deal with this prediction problem. However, several works in literature have addressed the promoter prediction problem using datasets containing sequences of 250 nucleotides or more. As the sequence length defines the amount of dataset attributes, even considering a limited number of properties to characterize the sequences, datasets with a high number of attributes are generated for training classifiers. Once high-dimensional datasets can degrade the classifiers predictive performance or even require an infesible processing time, predicting promoters by training classifiers from datasets with a reduced number of attributes, it is essential to obtain good predictive performance with low computational cost. To the best of our knowledge, there is no work in literature that verified in a sistematic way the relation between the sequences length and the predictive performance of classifiers. Thus, in this work, sixteen datasets composed of different sized sequences are built and evaluated using the SVM and k-NN classifiers. The experimental results show that several datasets composed of shorter sequences acheived better predictive performance when compared with datasets composed of longer sequences and consumed a significantly shorter processing time.

1 Introduction

Over recent years, advances in technology have allowed an acceleration of new genomes sequencing [9], evidencing the increasing demand for data analysis automation and for improving procedures previously used [2]. This has encouraged studying and implementing several computational techniques and creating new tools to enable processing of large amounts of genomic data.

One of the first steps for functional genomic annotation is promoter identification. Promoters are regions responsible for signaling and controlling the exact

* This research was partially supported by CNPq, FAPEMIG and UFOP.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 41–48, 2014.

position where the transcription mechanism initiates, called TSS (*Transcription Start Site*). The capability for detecting them in their different forms will make it possible to understand how, where and when transcription occurs, in addition to providing clarification on the interaction network and regulation of the transcription mechanism [8,9].

The identification of promoter sequences in genomes can be seen as a classification problem, where, given the features of a genomic sequence, it would be classified as promoter or non-promoter. Therefore, several computational approaches to predict promoters have been proposed using different classification techniques and different types of information extracted from sequences. Nevertheless, further progress is needed to improve them [14,1,6,9].

Much of the complexity of promoter prediction problem is due to their diverse nature, which makes it difficult to identify them [12,8,10]. Therefore, a crucial step for prediction success is to discover features of promoter sequences that are relevant to differentiate them from non-promoter sequences.

In the search for relevant features to distinguish between promoter and non-promoter sequences, several properties of sequences have been tested for their predictive capability. According to [14], a prediction strategy can use three types of features: structural, based on signs and based on context. Several studies have shown that in order to build accurate models to predict or describe genomic processes, the structural properties of the DNA molecules must be considered [11]. Thus, the structural properties have been widely used in recent years [14] and have also been adopted for this work.

Despite the large amount of work involving promoter prediction [12,8,1,2,6,7,9], to the best of our knowledge, none of them verified in a systematic way the relation between the length of sequences used for training classification models and their predictive performance. Thus, the aim of this work is to evaluate, through the application of classification techniques, the effect of the sequence length in discrimination between promoters and non-promoters.

The importance of this evaluation is due to the fact that, considering the structural properties, the longer the sequences used to compose datasets used for training classifiers, the greater the amount of attributes. The problem is that high-dimensional datasets, that is, with great number of attributes, make the classification a more complex process, and the result may be an increase in classifiers training time and a reduction of their predictive perfomance.

Due to the amount of data available and the attention it has received from the scientific community in recent decades [14], the genome chosen to be studied in this work was *Homo sapiens*. The experiments were conducted using a well known and reliable promoter database which is publicly available on the web.

2 Our Approach

For the studies conducted in this work, promoter and non-promoter sequences derived from human genome were used for datasets construction.

Promoters were obtained from a set of sequences available in the DBTSS database [13], version 8.0. DBTSS, which has already been used in several other works [6,8,9,2], is a set of approximately 98,000 experimentally validated promoter sequences with active TSS, where each sequence has 1201 bp (base pairs).

Non-promoters correspond to several genomic sequences that were extracted randomly from intergenic regions and from introns and exons [6]. The criteria for obtaining these sequences require that the region is at a minimum distance of 1000 nucleotides from the positions demarcated on CAGE database, indicating transcription regions, and at a minimum distance of 1000 nucleotides from the positions demarcated on RefSeq that has information denoting the beginning of genes. Thus, the selection of false non-promoter sequences is avoided. CAGE and RefSeq databases were obtained from *pppBenchmark* tool [16] website[1].

Due to computational cost to process a sequence dataset, only part of the sequences available at DBTSS database was used in the composition of the datasets of this study. Thus, a total of 7000 different promoter sequences were chosen randomly, avoiding the inclusion of noisy sequences. In addition, other 7000 non-promoter sequences complete the datasets.

Therefore, all datasets used in this work are composed of the same 14000 sequences. However, the length of sequences varies from one dataset to another. For example, the dataset called 250-50 consists of sequences represented by 301 nucleotides. For promoter sequences, this size is the sum of the number of nucleotides positioned upstream and downstream of TSS (in addition to TSS itself), that is, in the example there are 250 nucleotides upstream and 50 nucleotides downstream of TSS. Therefore, for the same dataset, TSS is always located at the same position in all promoter sequences. Since non-promoter sequences do not have TSS, their length is simply given by their number of nucleotides. Thus, in 250-50 dataset, non-promoter sequences are also composed of 301 nucleotides.

Each dataset sequence is characterized by a set consisting of 13 structural properties [11], named: A-philicity, base stack energy, B-DNA, bendability, DNA-bending stiffness, disrupt energy, DNA denaturation, free energy, nucleosome positioning, propeller twist, protein deformation, protein-DNA twist and Z-DNA. These properties, which have already been subject of other studies in literature [7,1,9], are physico-chemical and conformational properties.

Since the structural properties may be determined by local interactions among neighboring nucleotides in a sequence [11], they are represented by tables where each possible nucleotide combination is associated with a value that represents its contribution to a particular structural property. As an example, Table 1 presents the mapped values of oligonucleotides for the stacking energy structural property.

Using these 13 structural properties, each nucleotide sequence (promoters and non-promoters) is converted into a numerical vector that characterizes it. Figure 1 illustrates the conversion of a sequence to two structural properties (protein deformation and nucleosome positioning). As it can be observed, the numerical vector of each property (structural profile) is obtained from scanning the sequence of nucleotides where, depending on the structural property, each vector

[1] Available at `http://bioinformatics.psb.ugent.be/webtools/pppbenchmark/`

Table 1. Mapped values of oligonucleotides for *base stack energy* property [3]

Oligonucleotides	Value (kcal/mole)	Oligonucleotides	Value (kcal/mole)
AA	-5.37	GA	-9.81
AC	-10.51	GC	-14.59
AG	-6.78	GG	-8.26
AT	-6.57	GT	-10.51
CA	-6.57	TA	-3.82
CC	-8.26	TC	-9.81
CG	-9.69	TG	-6.57
CT	-6.78	TT	-5.37

value is obtained considering sequences of dinucleotides (protein deformability) or trinucleotides (nucleosome positioning).

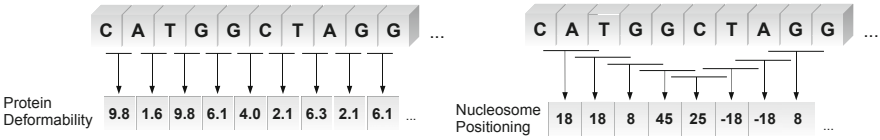


Fig. 1. Conversion of a sequence to two structural properties

Considering the conversion schema previously mentioned, in order to show the capability of the structural properties to discriminate promoter from non-promoter sequences, Figure 2 illustrates, for two structural properties, the average structural profile of promoter and non-promoter sequences of the 250-50 dataset. In this figure, TSS is located at the 0 position.

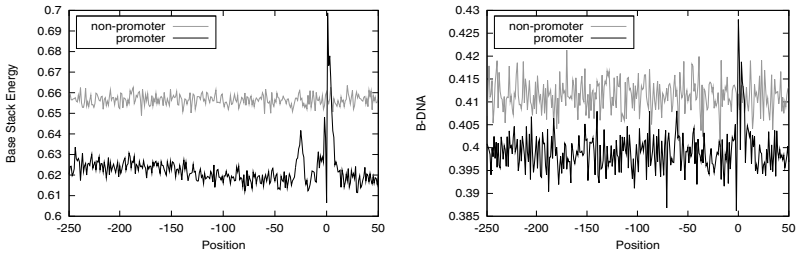


Fig. 2. Structural profiles for the *250-50* dataset

The complete characterization of a sequence is given by a single numerical vector resulting from the junction of the vectors representing each of the 13 structural properties considered in this work. The size of the resultant vector of these junctions corresponds to the number of predictor attributes of the datasets

used for classifiers training. In addition to these predictor attributes, each sequence has a value for the class attribute, which indicates whether that sequence is promoter or non-promoter. As an example, the largest dataset used in our experiments, the 250-50 one, results in a set of 3898 predictor attributes. Table 2 shows the number of predictor attributes for each dataset used in this work.

Table 2. Predictor attributes for each dataset

Dataset	Number of atributes	Dataset	Number of atributes
10-1	141	20-50	908
10-3	167	30-50	1038
10-5	193	40-50	1168
10-10	258	50-50	1298
10-20	388	100-50	1948
10-30	518	150-50	2598
10-40	648	200-50	3248
10-50	778	250-50	3898

As it can be observed in Table 2, the length of sequences used to compose the dataset defines the amount of their attributes. Several studies in literature have addressed the problem of promoter prediction using datasets containing sequences of 250 nucleotides or more [12,2,8,9]. Although a limited amount of features is being used in characterization of sequences, high-dimensional datasets are generated for classifiers training.

The problem with high-dimensional datasets, that is, with high number of attributes, is that they make classification a more complex process, often consuming an infeasible time for training classifiers and degrading their predictive performance.

Therefore, to predict promoters by training classifiers from datasets with a reduced number of attributes, it is essential to obtain good predictive performance with low computational cost. This way, the objective of the experiments conducted in this work is to evaluate the impact of the sequence length variation on the classifiers performance.

3 Computational Experiments

3.1 Classifiers and Experimental Setup

SVM (Support Vector Machine) and k-NN (k-Nearest Neighbours) classifiers, usually adopted in data mining works, were chosen to evaluate the impact of the sequence length variation on the performance of predictive models. Experiments were conducted using the caret package (short for *classification and regression training*) in R [15], which is a programming language and an environment widely used in statistical and graphics computation for data analysis.

k-NN classifier's idea is very simple. Each dataset instance is described by an n-dimensional vector, where k corresponds to the number of predictor attributes.

To classify a new instance (an instance whose class is unknown), the classifier uses distance metrics to determine the k training instances that are more similar to the instance to be classified. Then, the most frequent class among the similar k instances is attributed to the new instance. In k-NN, the k value is an input parameter.

Considering each dataset instance as a point in n-dimensional space, the basic idea of SVM is to find a hyperplane with maximum margin of separation, ie, one that provides the separation of training instances, with maximum margin, in two portions in n-dimensional space. Once the optimal hyperplane is found, the classification of a new instance is made by determining its position in relation to the separation hyperplane. Although this method was originally proposed for binary classification problems, several extensions have been proposed in literature to make it suitable for multi-class classification problems.

In order to set the algorithms parameters for the dataset used in this study, experiments were conducted by varying the parameters values C (0.25, 0.5, 1, 2, 4), *gamma*([0.1, 0.0001]), for SVM (using RBF kernel) and k (1, 3, 5, 7, 9) for k-NN. Table 3 presents the best parameter values obtained for each dataset and therefore used in our experiments to obtain the results presented here. All experiments were carried out on a Core i7-2600 @ 3.40GHz PC with 12 GBytes of RAM.

Table 3. k-NN e SVM parameters

	k-NN	SVM	
Dataset	k	C	*gamma*
10-1	9	1	3.64e-03
10-3	9	0.5	3.05e-03
10-5	9	0.5	1e-02
10-10	9	2	1e-03
10-20	9	1	1e-03
10-30	9	1	1e-03
10-40	9	1	7.84e-04
10-50	9	1	1e-03

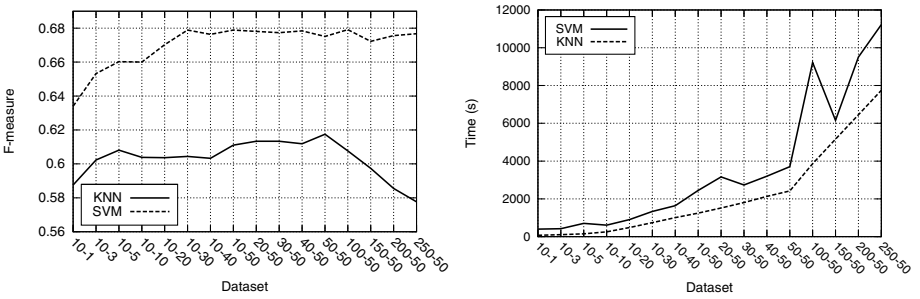
	k-NN	SVM	
Dataset	k	C	*gamma*
20-50	9	1	1e-03
30-50	9	0.5	1e-03
40-50	9	0.5	1e-03
50-50	7	0.5	1e-03
100-50	9	1	1e-03
150-50	9	0.5	1.96e-04
200-50	9	1	1.56e-04
250-50	9	1	1e-04

3.2 Experimental Results

The classifiers predictive performance was measured using k-cross-validation (k =10) and F-measure metric. For each dataset, the same test partitions were used in the evaluation of classifiers.

The results of the experiments are presented in Figure 3 graphs. Figure 3(a) graph shows the predictive performance of SVM and k-NN classifiers for each of the 16 datasets evaluated. Figure 3(b) graph shows the processing time spent in the classification process for these datasets.

As it can be seen in Figure 3(a) graph, the SVM classifier obtained better predictive performance than the k-NN one for all datasets evaluated.



(a) (b)

Fig. 3. (a) Average F-measure and (b) processing time

Yet, the most important thing to observe in Figure 3(a) graphs is that for both classifiers, the decrease in the length of sequences used in the datasets did not necessarily imply a reduction in their predictive performance. SVM performance remained relatively stable for datasets composed of sequences ranging in size from 301 (250-50) to 41 (10-30) nucleotides, presenting a marked degradation in performance only for sequences containing less than 41 nucleotides. k-NN achieved its best performance with the 50-50 dataset and, even for the dataset composed of shorter sequences (10-1), presented superior predictive performance compared with larger datasets (250-50).

Figure 3(b) graph shows that, for both classifiers, time spent for processing datasets grows exponentially with the increase of the length of sequences that compose them. It is worth noting that in many cases, a dataset composed of shorter sequences achieves superior predictive performance compared with longer sequence datasets and time spent in processing is significantly shorter than that consumed by longer sequence datasets. For example, for SVM, the 10-30 dataset presents predictive performance slightly higher than that achieved by the 250-50 dataset and time spent in processing is more than 8 times shorter than that spent by the 250-50 dataset.

4 Conclusion

Promoter prediction is a fundamental step for genome functional annotation and, therefore, several computational approaches have been proposed using different classification techniques. However, to best of our knowledge, none of them verified in a systematic way the relation between the length of sequences used for training classification models and their predictive performance. This way, experiments were conducted to analyze the impact of the sequence length variation on the classifiers performance.

In order to perform the analysis previously mentioned, 16 datasets composed of different sized sequences were generated and evaluated using the SVM and k-NN classifiers. The experimental results show that the decrease in the length

of sequences used in the composition of the datasets did not necessarily result in a reduction of the classifiers predictive performance. In addition, several bases composed of shorter sequences achieved superior predictive performance compared with datasets composed of longer sequences and consumed a significantly shorter processing time.

As future work, we plan to apply techniques for selecting attributes in datasets generated in this study aiming at reducing the datasets number of attributes and improving classifiers predictive performance.

References

1. Abeel, T., Saeys, Y., Bonnet, E., Rouzé, P., Van de Peer, Y.: Generic eukaryotic core promoter prediction using structural features of dna. Genome Research 18(2), 310–323 (2008)
2. Abeel, T., Saeys, Y., Rouzé, P., Van de Peer, Y.: Prosom: core promoter prediction based on unsupervised clustering of dna physical profiles. Bioinformatics 24(13), i24–i31 (2008)
3. Baldi, P., Brunak, S., Chauvin, Y., Pedersen, A.G.: Computational applications of dna structural scales. In: Glasgow, J.I., Littlejohn, T.G., Major, F., Lathrop, R.H., Sankoff, D., Sensen, C. (eds.) ISMB, pp. 35–42. AAAI (1998)
4. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
5. Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
6. Dineen, D., Schroder, M., Higgins, D., Cunningham, P.: Ensemble approach combining multiple methods improves human transcription start site prediction. BMC Genomics 11(1), 677 (2010)
7. Florquin, K., Saeys, Y., Degroeve, S., Rouzé, P., Van de Peer, Y.: Large-scale structural analysis of the core promoter in mammalian and plant genomes. Nucleic Acids Research 33(13), 4255–4264 (2005)
8. Gan, Y., Guan, J., Zhou, S.: A pattern-based nearest neighbor search approach for promoter prediction using dna structural profiles. Bioinf. 25(16), 2006–2012 (2009)
9. Gan, Y., Guan, J., Zhou, S.: A comparison study on feature selection of dna structural properties for promoter prediction. BMC Bioinformatics 13(1), 4 (2012)
10. Grishkevich, V., Hashimshony, T., Yanai, I.: Core promoter t-blocks correlate with gene expression levels in c. elegans. Genome Research 21(5), 707–717 (2011)
11. Meysman, P., Marchal, K., Engelen, K.: DNA structural properties in the classification of genomic transcription regulation elements. Bioinformatics and Biology Insights 6, 155–168 (2012)
12. Ohler, U., Niemann, H., Liao, G.C., Rubin, G.M.: Joint modeling of dna sequence and physical properties to improve eukaryotic promoter recognition. Bioinformatics 17(suppl. 1), S199–S206 (2001)
13. Yamashita, R., Sugano, S., Suzuki, Y., Nakai, K.: Dbtss: Database of transcriptional start sites progress report in 2012. Nucleic Acids Res. 40(D1), 150–154 (2012)
14. Zeng, J., Zhu, S., Yan, H.: Towards accurate human promoter recognition: a review of currently used sequence features and classification methods. Briefings in Bioinformatics 10(5), 498–508 (2009)
15. Kuhn, M., Johnson, K.: Applied Predictive Modeling. Springer (2013)
16. Abeel, T., Van de Peer, Y., Saeys, Y.: Toward a gold standard for promoter prediction evaluation. Bioinformatics 25(12), i313–i320 (2009)

Evolution of Genes Neighborhood within Reconciled Phylogenies: An Ensemble Approach

Cedric Chauve[1], Yann Ponty[1,2], and João Paulo Pereira Zanetti[1,3,4]

[1] Department of Mathematics, Simon Fraser University, Burnaby, Canada
[2] Pacific Institute for Mathematical Sciences, CNRS UMI3069, Vancouver, Canada
[3] Institute of Computing, UNICAMP, Campinas, Brazil
[4] São Paulo Research Foundation, FAPESP, São Paulo, Brazil

Abstract. We consider a recently introduced dynamic programming scheme to compute parsimonious evolutionary scenarios for gene adjacencies. We extend this scheme to sample evolutionary scenarios from the whole solution space under the Boltzmann distribution. We apply our algorithms to a dataset of mammalian gene trees and adjacencies, and observe a significant reduction of the number of syntenic inconsistencies observed in the resulting ancestral gene adjacencies.

1 Introduction

The reconstruction of the evolutionary history of genomic characters along a given species tree is a long-standing problem in computational biology. This problem has been well studied for several types of genomic characters, for which efficient algorithms exist to compute parsimonious evolutionary scenarios; classical examples include the case of genes and genomes sequences [9], gene content [5], and gene family evolution [1,8]. Recently, Bérard *et al.* [3] extended the corpus of such results to syntenic characters, as they introduced the notion of adjacency tree, that models the evolution of gene adjacencies within a phylogeny, motivated by the reconstruction of the architecture of ancestral genomes and described an efficient dynamic programming (DP) algorithm, called `DeCo`, to compute parsimonious adjacency evolutionary histories.

From a methodological point of view, most existing algorithms to reconstruct evolutionary scenarios along a species tree in a parsimony framework rely on some dynamic-programming along this tree, whose introduction can be traced back to Sankoff in the 1970s for parsimony-based models (see [6] for a recent retrospective on this topic). Recently, several works have explored more general approaches for such parsimony problems that either explore a wider range of values for combinatorial parameters of parsimonious models [11,10] or consider several alternate histories for a given instance, chosen for example from the set of all possible co-optimal scenarios (see [2,14] for examples of this approach for the gene tree/species tree reconciliation problem), or from the whole solution space, thus including suboptimal solutions [7].

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 49–56, 2014.

The present work follows the later approach and extends the DeCo DP scheme toward an exploration of the whole solution space of adjacency histories, under the Boltzmann probability distribution, that assigns a probability to each solution defined in terms of its parsimony score. This principle of exploring the solution space of a combinatorial optimization problem under the Boltzmann probability distribution, has been applied in several contexts and is sometimes known as the "Boltzmann ensemble approach" (see [13] for an illustration on RNA secondary structure prediction). While this Boltzmann ensemble approach has been used for a long time in RNA structure analysis, to the best of our knowledge it is not the case in comparative genomics, where exact probabilistic models have been favored as increasing computational capacities allow them to handle realistic datasets. However, such a probabilistic model does not exist so far for gene adjacencies, which motivates our work.

In the present paper, we modify the DP scheme of DeCo in order to sample gene adjacencies histories under the Boltzmann distribution. We apply our sampling algorithm to the dataset of over $6,000$ pairs of mammalian gene trees considered in the original DeCo paper [3]. This dataset was characterized by a significant number of ancestral genes involved in more than 2 adjacencies, which correspond to syntenic inconsistencies. We show that by sampling adjacencies histories under a Boltzmann distribution that favors co-optimal histories and conserving only frequent ancestral adjacencies, we can reduce significantly the number of syntenic inconsistencies.

2 Models

A *phylogeny* is a rooted tree which represents the evolutionary relationships of a set of elements represented by its nodes: internal nodes are ancestors, leaves are extant elements, and edges represent direct descents between parents and children. We consider here three kinds of phylogenies (illustrated in Figure 1): species trees, reconciled gene trees and adjacencies trees/forests. Trees we consider are always rooted. For a tree T and a node x of T, we denote by $T(x)$ the subtree rooted at x. If x is an internal node, we assume it has either one child, denoted by $a(x)$, or two children, denoted by $a(x)$ and $b(x)$. A tree where all internal nodes have two children is called a *binary tree*.

Species Trees. A species tree S is a binary tree that describes the evolution of a set of related species, from a common ancestor (the root of the tree), through the mechanism of *speciation*. For our purpose, species are identified with *genomes*, and genes are arranged linearly or circularly along chromosomes.

Reconciled Gene Trees. A reconciled gene tree is also a binary tree that describes the evolution of a set of genes, called a *gene family*, through the evolutionary mechanisms of speciation, *gene duplication* and *gene loss*, within the given species tree S. Therefore, each node of a gene tree G represents a gene loss, an extant gene or an ancestral gene. Ancestral genes are represented by the internal nodes of G, while gene losses and extant genes are represented by the leaves of G.

We denote by $s(g) \in S$ the species of a gene $g \in G$, and by $e(g)$ the evolutionary event that leads to the creation of the two children $a(g)$ and $b(g)$. If g is an internal node of G, then $e(g)$ is a speciation (denoted by Spec) if the species pair $\{s(a(g)), s(b(g))\}$ equals the species pair $\{a(s(g)), b(s(g))\}$, or a gene duplication (GDup) if $s(a(g)) = s(b(g)) = s(g)$. Finally, if g is a leaf, then $e(g)$ indicates either a gene loss (GLoss) or an extant gene (Extant), in which case $e(g)$ is not an evolutionary event.

Adjacency Trees and Forests. A *gene adjacency* is a pair of genes that appears consecutively along a chromosome. An adjacency tree represents the evolution of an ancestral adjacency through the evolutionary events of speciation, gene duplication, gene loss (these events, as described above, occur at the gene level and are modeled in the gene trees), and *adjacency duplication* (ADup), *adjacency loss* (ALoss) and *adjacency break* (ABreak), that are adjacency-specific events.

- The duplication of an adjacency $\{g_1, g_2\}$ follows from the simultaneous duplication of both its genes g_1 and g_2 (with $s(g_1) = s(g_2)$ and $e(g_1) = e(g_2) =$ GDup), resulting in the creation of two distinct adjacencies each belonging to $\{a(g_1), b(g_1)\} \times \{a(g_2), b(g_2)\}$.
- An adjacency may disappear due to several events, such as the loss of exactly one (gene loss) or both (adjacency loss) of its genes, or a genome rearrangement that breaks the contiguity between the two genes (adjacency break).

Finally, to model the complement of an adjacency break, i.e. the creation of adjacencies through a genome rearrangement, *adjacency gain* (AGain) events are also considered, and result in the creation of a new adjacency tree. It follows that the evolution of the adjacency between two genes can be described by a forest of adjacency trees, called an *adjacency forest*. In this forest, each node v belongs to a species denoted by $s(v)$, and is associated an evolutionary event $e(v) \in \{\text{Spec}, \text{GDup}, \text{ADup}\}$ if g is an internal node, or $\{\text{Extant}, \text{GLoss}, \text{ALoss}, \text{ABreak}\}$ if v is a leaf. Finally, adjacency gain events are associated to the roots of the trees of the adjacency forest.

Parsimony Scores and the Boltzmann Distribution. When considered in a parsimonious framework, the score of a reconciled gene tree G can be either its duplication cost (number of gene duplications) or its mutation cost (number of gene duplications and losses). The score of an adjacency forest F is the number of adjacency gains and breaks; other events are not considered as they are the by-products of evolutionary events already accounted for in the score of the reconciled gene trees G_1 and G_2. We denote by $s_a(F)$ the parsimony score of an adjacency forest F. Let $\mathcal{F}(G_1, G_2)$ be the set of all adjacency forests for G_1 and G_2, including both optimal and sub-optimal ones, where we assume that at least one extant adjacency is composed of extant genes from G_1 and G_2. The *partition function* associated to G_1 and G_2 is defined by

$$\mathcal{Z}(G_1, G_2) = \sum_{F \in \mathcal{F}(G_1, G_2)} e^{-\frac{s_a(F)}{kT}}$$

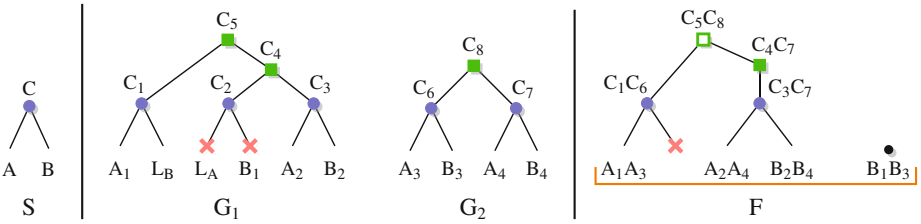


Fig. 1. (Left) Species tree S, with two extant species A and B and an ancestral species C. (Middle) Two reconciled gene trees G_1 and G_2, with four extant genes in genome A, four extant genes in genome B and three ancestral genes in genome C. The set of extant adjacencies is $(A_1A_3, B_1B_3, A_2A_4, B_2B_4)$ (Right) Parsimonious adjacency forest F composed of two adjacency trees. Blue dots are speciation nodes. Leaves are extant (species, genes, adjacencies), except when labeled with a red cross (gene loss). Green squares are (gene or adjacency) duplication nodes. Gene labels refer to the species of nodes. Every node of the adjacency tree is labeled by a couple of nodes from gene trees. Figure adapted from [3].

where kT is an arbitrary constant. The partition function implicitly defines a *Boltzmann probability distribution* over $\mathcal{F}(G_1, G_2)$, where the probability of an adjacency forest F is defined by:

$$P(F) = \frac{e^{\frac{-s_a(F)}{kT}}}{\mathcal{Z}(G_1, G_2)}.$$

By exponentially favoring adjacency forests with lower parsimony scores, the Boltzmann distribution provides an alternative way to probe the search space, which is heavily influenced by the choice of kT. Indeed, decreasing kT values will skew the Boltzmann distribution towards more parsimonious adjacency forests. Its limiting distributions are uniform over the whole search space ($kT \to +\infty$) or over the set of co-optimal forests ($kT \to 0$).

3 Algorithms

`DeCo`, the algorithm described in [3] to compute a parsimonious adjacency forest, is a DP scheme constrained by S, G_1 and G_2. We first present this algorithm, then describe how to turn it into a sampling algorithm.

The `DeCo` *DP Scheme.* Let G_1 and G_2 be two reconciled gene trees and g_1 and g_2 be two nodes, respectively of G_1 and G_2, such that $s(g_1) = s(g_2)$. The `DeCo` algorithm computes, for every such pair of nodes g_1 and g_2, two quantities denoted by $c_1(g_1, g_2)$ and $c_0(g_1, g_2)$, that correspond respectively to the most parsimonious score of a parsimonious adjacency forest for the pairs of subtrees $G(g_1)$ and $G(g_2)$, under the hypothesis of a presence (c_1) or absence (c_0) of an ancestral adjacency between g_1 and g_2. As usual in DP along a species

tree, the score of a parsimonious adjacency forest for G_1 and G_2 is given by $\min(c_1(r_1, r_2), c_0(r_1, r_2))$ where r_1 is the root of G_1 and r_2 the root of G_2.

So, $c_1(g_1, g_2)$ and $c_0(g_1, g_2)$ can be computed as the minimum of a sum of the scores of adjacency gains or breaks and, more importantly, of terms of the form $c_1(x, y)$ and $c_0(x, y)$ with $(x, y) \in \{g_1, a(g_1), b(g_1)\} \times \{g_2, a(g_2), b(g_2)\} - (g_1, g_2)$, using the two combinatorial operator min and $+$.

(Un)-ambiguity of the `DeCo` *DP Scheme.* As defined in [13], the ambiguity of a DP algorithm can be defined as follows: a DP explores a combinatorial solution space (here for `DeCo`, the space of all possible adjacency forests, including possible suboptimal solutions), that can be explicitly generated by replacing in the equations min by $\Cup$ (the set-theoretic union operator) and $+$ by the Cartesian product $\times$ between combinatorial sets. A DP algorithm is then unambiguous if the unions are disjoint, i.e. the sets provided as its arguments do not overlap.

Claim. The `DeCo` dynamic programming scheme is unambiguous.

Proof (Sketch). Computing $c_1(g_1, g_2)$ and $c_0(g_1, g_2)$ branches on disjoint subcases that each involve a different set of terms $c_1(x, y)$ and $c_0(x, y)$. The only case that deserves a closer attention is the case where $e(g_1) = e(g_2) = \mathsf{GDup}$, as a simultaneous duplication can be obtained by two successive duplications. But in this case, the number of AGain events is different (we refer the reader to the original `DeCo` equations [3]), which ensures the obtained solutions are different.

Stochastic Backtrack Algorithm through Algebraic Substitutions. As mentioned in [13], any unambiguous dynamic programming scheme, and in particular that of `DeCo`, can be modified to not only exhaustively generate the set of all adjacency forests, but also to compute the corresponding partition function by replacing the arithmetic operators $(\min, +) \to (\sum, \times)$, and exponentiating atomic costs $C \to e^{-C/kT}$.

Also, and more importantly for our purpose, it allows to sample adjacency forests under the Boltzmann distribution, by changing the deterministic backtrack used for maximum parsimony into a stochastic operation. Indeed, assume that the partition function version of the `DeCo` equation computes $c_1(g_1, g_2)$ (resp. $c_0(g_1, g_2)$) as $\sum_{i \in [1, k_1]} t_i$, where the t_i denote the contribution to the partition function of one of the local alternatives within the DP scheme. The latter are typically computed recursively as combinations of atomic adjacency gain/break costs, and *recursive* terms of the form $c_1(x, y)$ and $c_0(x, y)$ with $(x, y) \in \{g_1, a(g_1), b(g_1)\} \times \{g_2, a(g_2), b(g_2)\} - \{(g_1, g_2)\}$.

Then a (possibly non-parsimonious) solution can be built recursively for $c_1(g_1, g_2)$ (resp. $c_0(g_1, g_2)$), by branching on some t_i with probability $t_i / c_1(g_1, g_2)$ (resp. $t_i / c_0(g_1, g_2)$), and proceed recursively on each occurrence of a *recursive* term within the alternative t_i. The correctness of the algorithm, i.e. the fact that the random processes induces a Boltzmann distribution on adjacency forests, follows readily from general considerations on unambiguous DP schemes [13].

The stochastic nature of the backtrack does not affect its worst-case complexity. Thus our Boltzmann sampling algorithm, for an instance composed of two

gene trees G_1 and G_2 of respective sizes (number of leaves) n_1 and n_2, has time complexity of $\mathcal{O}(n_1 \times n_2)$ for each backtrack.

4 Experiments

Data. We re-analyzed a dataset described in [3] composed of $5,039$ reconciled gene trees and $50,389$ extant gene adjacencies, forming $6,074$ DeCo instances, with genes taken from 36 mammalian genomes from the Ensembl database in 2012. In [3], these data were analyzed using the `DeCo` algorithm that computed a single parsimonious adjacency forest per instance. All together, these adjacency forests defined $112,188$ (resp. $96,482$) ancestral and extant genes (reps. adjacencies) and, more important, lead to $5,817$ ancestral genes participating to three or more ancestral adjacencies, which represent a significant level of syntenic inconsistency (close to 5%), as a gene can only be adjacent to at most two neighboring genes along a chromosome.

Boltzmann Sampling. For each instance, we sampled $1,000$ adjacency forests under the Boltzmann distribution, for three values of kT, 0.001, 0.1, 0.5. The results were very similar for values 0.1 and 0.001, so we will not discuss the later. The main difference between the results obtained with $kT = 0.1$ and $kT = 0.5$ is that in the later case, non-optimal adjacency forests have a higher probability in the Boltzmann distribution, and thus are more likely to be sampled, while $kT = 0.1$ skews the distribution toward optimal adjacency forests.

In order to assess the impact of considering alternate, possibly non-optimal, adjacency forests, we computed, for each ancestral adjacency observed in the sampled adjacency forests, its frequency, defined as the number of adjacency forests it is observed in divided by $1,000$, that is the number of sampled adjacency forests. Then, we considered 10 frequency thresholds, from 0.1 to 1 by steps of 0.1, and looked at the numbers of ancestral adjacencies, genes and syntenic inconsistencies from ancestral adjacencies whose frequency is at least the chosen threshold. Table 1 below summarizes our main observations.

Discussion. We can deduce two main points from the results summarized in Table 1. First, the difference observed between the results obtained in the two experiments clearly suggest that parsimony is an appropriate criterion for looking at gene adjacency evolution. Indeed, in the results obtained with $kT = 0.5$, that gives a higher probability to non-optimal adjacency forests, it appears that the number of conserved ancestral adjacencies drops sharply after frequency 0.6, showing that very few ancestral adjacencies appear with high frequency. Next, with $kT = 0.1$, we can observe that by taking a high frequency (starting at a frequency threshold of 0.6), we reduce significantly the number of syntenic inconsistency while maintaining a relatively similar number of ancestral genes than the experiments described in [3]; this observation is our main finding, and illustrates the interest of the ensemble approach compared to the classical dynamic approach that relies on a single arbitrary optimal solution.

Table 1. Characteristics of ancestral genes and adjacencies from observed ancestral adjacencies filtered by frequency of observation (leftmost column), with values $kT = 0.1,\ 0.5$ (left / right)

Adjacency freq.	Ancestral genes	Ancestral adjacencies	Syntenic inconsistencies
≥ 0.1	120,269 / 122,367	116,204 / 136,480	14,851 / 31,221
≥ 0.2	119,540 / 121,657	113,265 / 130,316	12,479 / 26,230
≥ 0.3	118,687 / 120,631	110,180 / 121,307	10,351 / 19,369
≥ 0.4	117,639 / 118,074	106,973 / 110,649	8,330 / 11,336
≥ 0.5	116,231 / 112,812	103,479 / 99,672	6,677 / 5,522
≥ 0.6	114,538 / 104,703	99,720 / 88,270	5,113 / 2,868
≥ 0.7	112,564 / 92,449	96,039 / 74,482	4,092 / 1,256
≥ 0.8	110,086 / 75,206	91,821 / 58,214	3,276 / 424
≥ 0.9	107,564 / 45,702	87,790 / 33,648	2,710 / 33
$= 1$	100,443 / 16	79,078 / 8	1,348 / 0

5 Conclusion and Perspectives

The main contribution of our work is an extension of the `DeCo` dynamic programming scheme to sample adjacency forests under the Boltzmann distribution. The application of our sampling algorithm on a mammalian genes dataset, together with a simple, threshold-based, approach to filter ancestral adjacencies, proved to be effective to reduce significantly the number of syntenic inconsistencies when sampling co-optimal adjacency forests, illustrating the interest of the ensemble approach. This preliminary work raises several questions and can be extended along several lines.

Sampling is obviously a non-exact way to estimate ancestral adjacency frequencies. Ideally, for each pair (g_1, g_2) of ancestral genes from the same species in an instance, we would like to compute its exact frequency as an ancestral adjacency under the Boltzmann distribution, which corresponds to the Boltzmann probability of this feature. These probabilities can be computed exactly using a modification of the DP scheme, at no extra computational cost using the technique known as "inside/outside" algorithm. Similarly, the Boltzmann probabilities of the adjacency gains and breaks associated to ancestral adjacencies can be computed and then used to compute a *Maximum Expected Accuracy* adjacency forest, which is a parsimonious adjacency forest in a scoring model where each event is weighted by Boltzmann probability (see [4] for an example of this approach for RNA secondary structures).

Finally, we considered here an evolutionary model based on speciation, duplication and loss. A natural extension would be to include the event of lateral gene transfer in the model. Efficient reconciliation algorithms exist for several variants of this model [1,8], together with an extension of `DeCo`, called `DeCoLT` [12]. `DeCoLT` is also based on dynamic programming, and it is likely that the techniques we developed in the present work also apply to this algorithm, and that the open question discussed above are of interest for this model.

References

1. Bansal, M.S., Alm, E.J., Kellis, M.: Efficient algorithms for the reconciliation problem with gene duplication, horizontal transfer and loss. Bioinformatics 28(12), 283–291 (2012)
2. Bansal, M.S., Alm, E.J., Kellis, M.: Reconciliation revisited: Handling multiple optima when reconciling with duplication, transfer, and loss. Journal of Computational Biology 20(10), 738–754 (2013)
3. Bérard, S., Gallien, C., Boussau, B., Szöllosi, G.J., Daubin, V., Tannier, E.: Evolution of gene neighborhoods within reconciled phylogenies. Bioinformatics 28(18), 382–388 (2012)
4. Clote, P., Lou, F., Lorenz, W.A.: Maximum expected accuracy structural neighbors of an RNA secondary structure. BMC Bioinformatics, 13(S-5), S6 (2012)
5. Csűrös, M.: Ancestral reconstruction by asymmetric wagner parsimony over continuous characters and squared parsimony over distributions. In: Nelson, C.E., Vialette, S. (eds.) RECOMB-CG 2008. LNCS (LNBI), vol. 5267, pp. 72–86. Springer, Heidelberg (2008)
6. Csürös, M.: How to infer ancestral genome features by parsimony: Dynamic programming over an evolutionary tree. In: Models and Algorithms for Genome Evolution, pp. 29–45. Springer (2013)
7. Doyon, J.-P., Hamel, S., Chauve, C.: An efficient method for exploring the space of gene tree/species tree reconciliations in a probabilistic framework. IEEE/ACM Trans. Comput. Biology Bioinform. 9(1), 26–39 (2012)
8. Doyon, J.-P., Scornavacca, C., Gorbunov, K.Y., Szöllősi, G.J., Ranwez, V., Berry, V.: An efficient algorithm for gene/Species trees parsimonious reconciliation with losses, duplications and transfers. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 93–108. Springer, Heidelberg (2010)
9. Liberles, D.A. (ed.): Ancestral Sequence Reconstruction. Oxford University Press (2007)
10. Libeskind-Hadas, R., Wu, Y.-C., Bansal, M.S., Kellis, M.: Pareto-optimal phylogenetic tree reconciliation. Bioinformatics 30(12), 87–95 (2014)
11. Pachter, L., Sturmfels, B.: Parametric inference for biological sequence analysis. PNAS 101(46), 16138–16143 (2004)
12. Patterson, M., Szöllosi, G.J., Daubin, V., Tannier, E.: Lateral gene transfer, rearrangement, reconciliation. BMC Bioinformatics, 14(S-15), S4 (2013)
13. Ponty, Y., Saule, C.: A combinatorial framework for designing (Pseudoknotted) RNA algorithms. In: Przytycka, T.M., Sagot, M.-F. (eds.) WABI 2011. LNCS, vol. 6833, pp. 250–269. Springer, Heidelberg (2011)
14. Scornavacca, C., Paprotny, W., Berry, V., Ranwez, V.: Representing a set of reconciliations in a compact way. J. Bioinformatics and Computational Biology 11(2) (2013)

Dynamic Programming for Set Data Types

Christian Höner zu Siederdissen[1], Sonja J. Prohaska[2], and Peter F. Stadler[2]

[1] Dept. Theoretical Chemistry, Univ. Vienna, Währingerstr. 17, Wien, Austria
[2] Dept. Computer Science, and Interdisciplinary Center for Bioinformatics, Univ. Leipzig, Härtelstr. 16-18, Leipzig, Germany

Abstract. We present an efficient generalization of algebraic dynamic programming (ADP) to unordered data types and a formalism for the automated derivation of outside grammars from their inside progenitors. These theoretical contributions are illustrated by ADP-style algorithms for shortest Hamiltonian path problems. These arise naturally when asking whether the evolutionary history of an ancient gene cluster can be explained by a series of local tandem duplications. Our framework makes it easy to compute Maximum accuracy solutions, which in turn require the computation of the probabilities of individual edges in the ensemble of Hamiltonian paths. The expansion of the Hox gene clusters is investigated as a show-case application. For implementation details see `http://www.bioinf.uni-leipzig.de/Software/setgram/`

Keywords: formal grammar, outside grammar, dynamic programming, Haskell, Hamiltonian path problems, tandem duplications, Hox clusters.

1 Introduction

Dynamic Programming (DP) over rich index sets provides solutions of a surprising number of problems in discrete mathematics. Even for NP-hard problems such as the Travelling Salesman Problem (TSP) exact solutions can be obtained for moderate size problems of practical interest. The corresponding algorithms, however, are usually specialized and use specific properties of the problem in an *ad hoc* manner that does not generalize particularly well.

Algebraic dynamic programming (ADP) [1] defines a high-level descriptive domain-specific language for dynamic programs over sequence data. The ADP framework allows extremely fast development even of quite complex algorithms by rigorously separating the traversal of the state space (by means of context free grammars), scoring (in terms of suitable algebras), and selection of desired solutions. The use of CFGs to specify the state space is a particular strength of ADP since it allows the user to avoid indices and control structures altogether, thereby bypassing many of the pitfalls (and bugs) of usual implementations. Newer dialects of ADP [2,3] provide implementations with a running time performance close to what can be achieved by extensively hand-optimized versions, while still preserving most of the succinctness and high-level benefits of the original ADP language. The key goal is to develop a framework that makes it easy to

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 57–64, 2014.

implement complex dynamic programs by combining small, simple components. A first step in this direction was the introduction of grammar products [4], which greatly simplifies the specification of algorithms for sequence alignments and related dynamic programming tasks that take multiple strings as input. The second and third steps are introduced in this work: a formal system for dynamic programming over unordered data types, together with the mechanistic derivation of Outside algorithms; both implemented in `ADPfusion` [2].

Sequence data is not the only type of data for which grammar-like dynamic programs are of interest. Inverse coupled rewrite systems (ICOREs) [5] allow the user to develop algorithms over both, sequence and tree-like data. While no implementation for these rewrite systems is available yet, they already simplify the initial development of algorithms. This is important in particular for tree-like data. Their non-sequential nature considerably complicates these algorithms. The grammar underlying the alignment of ncRNA family models with `CMCompare` [6], which simultaneously recurses over two trees, may serve as an example for the practical complications. There are compelling reasons to use DP approaches in particular when more information than just a single optimal solution is of interest. DP over sequences and trees readily allows the enumeration of all optimal solutions, and it offers generic ways to systematically investigate suboptimal solutions and to compute the probabilities of certain sub-solutions. Classified dynamic programming [7], furthermore, enables the simultaneous calculation of solutions with different class features via the evaluation algebra instead of constructing different grammars for each class. Two well-known examples for DP over sequences in computational biology for which these features are extensively used in practise are pairwise sequence alignment and RNA folding. Due to the tight space limits we relegate them to the Electronic Supplement.

A quite different classical example of DP is the Travelling Salesman Problem (TSP). It is easily stated as follows: given a set X of cities and a matrix $d : X \times X \to \mathbb{R}_+$ of (not necessarily symmetric) distances between them, one looks for the tour (permutation) π on X that minimizes the tour length $f(\pi) := d_{\pi(n),\pi(1)} + \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)}$. W.l.o.g., we may set $X = \{1, \ldots, n\}$ and anchor the starting point of a tour at $\pi(1) = 1$. The well-known (exponential-time) DP solution for the TSP [8,9] operates on "sets with an interface" $[A, i]$ representing the set of all tours starting in $1 \in A$, then visiting all other cities in A exactly once and ending in $i \in A$. The length of the shortest path of this type is denoted by $f([A, i])$. For an optimal tour we have $f([X \setminus \{i\}, i]) + f(\langle i, 1\rangle) \to \min$, where $f(\langle i, 1\rangle) = d_{1,i}$ is the length of the edge from i to 1. The $f([A, i])$ satisfy the recursions

$$f([A, i]) = \min_{j \in A} f([A \setminus \{i\}, j]) + f(\langle j, i\rangle) \tag{1}$$

since the shortest path through A to i must consist of a shortest path through A ending in some $j \in A$ and a final step from j to i. The fundamental question that we will address in this contribution is whether we can rephrase this and similar DP algorithms also in an ADP like manner. In other words: how can we separate state space traversal and evaluation, even though we do not have a grammar at hand (because we do not even operate on strings or ordered trees)?

2 ADP over Set-Like Data Types

Generalized Decompositions. The key observation is that we have to generalize the notion of *parsing a string* to much more general ways of traversing the state space. This interpretation of "productions" makes perfect sense for the paths in the TSP solution. The operator $+\!\!+$ provides the decomposition of the set A into A' as well as a terminal e denoting the newest edge added to A' to construct A.

$$\text{“}A \to A' +\!\!+ e\text{”} := \{[A,i] \mapsto [A \setminus \{i\}, j] +\!\!+ \langle j,i\rangle | A \subseteq X,\, j \in A\} \tag{2}$$

The path variables $[A,i]$ highlight a second important ingredient of the formalism. Each object $[A,i]$ consists of an interior part $\text{int}([A,i]) = A \setminus \{i,1\}$ and the interface $\partial A = \{1,i\}$. The latter consists of the vertices that need to be known explicitly for the evaluation: they will appear explicitly in the evaluation algebra. For fixed A in the production (2), e.g., we have to consider all $j \in A \setminus \{1\}$ as possible endpoints of the paths.

The distinction between interior and interface of each object $A := [\text{int}(A), \partial A]$ allows a more principled way to constructing concrete decompositions:

$$[\text{int}(A), \partial A] \mapsto \mathop{+\!\!+}_{i} [\text{int}(A_i), \partial A_i] \tag{3}$$

with the following properties:

(C1) $\bigcup_i A_i = A$, i.e., the parts of A form a covering of A.
(C2) $\text{int}(A_i) \cap \text{int}(A_j) \neq \emptyset$ implies $i = j$, i.e., the interiors of the parts are disjoint.
(C3) $\text{int}(A_i) \subseteq \text{int}(A)$, i.e., the interiors behave like isotonic functions.

The intuition behind axiom (C1) is that any decomposition of an object must eventually evaluate all parts. Condition (C2) and (C3) implies that the interiors of the parts can be evaluated independently. To allow meaningful evaluation algebras in the ADP sense we require that concatenation is associative. It may be tempting to think of ∂ and int in terms of generalized topological functions, i.e., as boundary and interior operators. This may not need to be the case in full generality, since we may have situations where A is not just a set.

A terminal is an object for which there is no further concrete decomposition. In the TSP examples, the terminals are on the one hand the edges $\langle j,i\rangle$ that appear explicitly in the decompositions as well as the path $\langle 1\rangle := [\{1\},1]$ of length 0 that appears implicitly as the base case of the concrete decompositions. The boundary ∂A is not necessarily just an unstructured set. For the asymmetric TSP, e.g., start and end point of a path $[A,j]$ are distinguished.

So far our discussion has been focussed on the decomposition of inputs in the terms of a grammar. The goal to optimize with an objective function in DP has only entered in passing, as in the TSP example in equ. (1). For DP to work, however, more is required. The grammar performs the decomposition of each subinput into its constituent elements, or terminal and non-terminal symbols. Each

of the different decompositions is then evaluated using an evaluation algebra that defines how $f(A)$ depends on the evaluation $f(A_i)$ of the fragments. In general there are multiple alternative decompositions of A. For the TSP for instance, we have to consider $A \to A \setminus \{i\}$ for all $i \in A$. It also is the job of the score algebra to combine the scores over these alternatives. To minimize f, scores are added over constituents and minimized over alternatives. To compute partition functions they are multiplied over constituents and added up over alternatives. Finally *Bellman's principle* [8] stipulates that decomposition and scoring play together in such a way that optimal solutions are always obtained by composing optimal solutions of smaller problems. We can implement algorithms specified in this formal system very efficiently (both in terms of programming effort and actual running times) using an extension to `ADPfusion`. Details are given in the Electronic Supplement.

Deriving Outside Algorithms. A key advantage of DP algorithms is the generic possibility to compute solutions with constraints, such as alignments that contain a given alignment edge. The basic idea behind this possibility is the combination of an "inside" with an "outside" solution, i.e., a pair of complementary partial solutions. Well known examples are pairwise sequence alignments or RNA folding. In the first example, pairwise alignments of prefixes are the objects of the forward (or "inside") recursion, while suffix-alignments are required as "outside" objects. In the RNA case, this is even more transparent, since "inside" runs over secondary structures on intervals, while the outside algorithm recurses over the complements of the intervals, again proceeding from smaller to larger outside objects. A good example is McCaskill's algorithm for computing the base pairing probabilities in the ensemble of all secondary structures formed by an input RNA molecule [10]. The construction of the outside traversal is a difficulty in ADP that has not been fully solved. For CFGs, thesis [11] shows that a grammar for the outside objects can be derived by doubling the input string and re-interpreting the region outside of interval $[i, j]$ as the interval $[j+1, i'-1]$ where i' is the equivalent position i in the 2nd copy of the input.

To make use of the full potential of dynamic programming it would be highly desirable to construct suitable outside traversals automatically from a given inside traversal. In the remainder of this section we discuss some of the general principles underlying the relationship of inside and outside recursion on a general level. The key observation is that the distinction of inside and outside comes from a generic way of splitting solutions so that

$$[\mathrm{int}(X), \partial A] \to [\mathrm{int}(A), \partial A] +\!\!+ [\mathrm{int}(A^*), \partial^* A^*] \tag{4}$$

corresponds to the set of all solutions that are constrained to $\partial A = \partial^* A^*$, i.e., that contain the particular feature specified by ∂A. Set-like objects have a straightforward explicit definition of their outside objects: $\mathrm{int}(A^*) := X \setminus (\mathrm{int}(A) \cup \partial A)$. The notation $\partial^* A^*$ emphasizes that in the case of structured interfaces corresponding inside and outside objects must consist of the same terminals, but possibly in different orderings. In the Electronic Supplement we illustrate this construction for RNA folding and pairwise alignments.

The straightforward definition of outside objects suggests that it should also be possible to construct inside-style productions for these outside objects in a generic, rule-based manner. It turns out that the solution to this long-standing problem in DP becomes surprisingly simple as soon as we allow ourselves to "parse" also data structures that are not strings or trees. With each concrete inside decomposition $A \mapsto (+\!\!+_i A_i) +\!\!+ (+\!\!+_j \langle t_j \rangle)$, where the A_i are non-terminals and the $\langle t_j \rangle$ are terminals, we associate

$$A_k^* \mapsto A^* +\!\!+ \left(\mathop{+\!\!+}_{i \neq k} A_i \right) +\!\!+ \left(\mathop{+\!\!+}_{j} \langle t_j \rangle \right) \tag{5}$$

For examples and a discussion of start non-terminals and empty terminals we refer to the Electronic Supplement. The situation is even simpler for the TSP: we have $[A, (1, i)]^* = [(X \setminus A) \setminus \{1, i\}, (i, 1)]$. In particular, $[A, (1, i)] +\!\!+ [A, (1, i)]^*$ corresponds to the set of all Hamiltonian paths that run from 1 to i through A and then from i back to 1 through $X \setminus A$. The same idea applies to other Hamiltonian path problems.

3 Application to Gene Cluster Histories

Local duplication of DNA segments via unequal crossover is the most plausible mechanism for the emergence and expansions of local clusters of evolutionary related genes. It remains hard and often impossible to disentangle the history of ancient gene clusters in detail even though polynomial-time algorithms exist to reconstruct duplication trees from pairwise evolutionary distance data [12]. The reason is the limited amount of phylogenetic information in a single gene. The situation is often aggravated by the extreme time scales leading to a decay of the phylogenetic signal so that only a few, very well-conserved sequence domains can be compared. A large number of trees then fits the data almost equally well. A meaningful analysis thus must resort to some form of summary that is less detailed than a duplication tree. In the absence of genome rearrangements, and if duplication events are restricted to copying single genes to adjacent positions, we expect phylogenetic distance to vary monotonically with genomic distance. A shortest Hamiltonian path through the phylogenetic distance matrix therefore should conform to the linear arrangement of the genes on the genome. The same high noise level that suggests to avoid duplication trees makes us distrust a single shortest path. Rather, we would like to obtain information on the ensemble of all Hamiltonian paths.

The shortest Hamiltonian path problem, well known to be NP-complete, is closely related to the TSP, and admits a similar dynamic programming solution [8,9]. We provide here an efficient implementation in our ADP-style framework. Denote by $[i, A, j]$ with $i, j \in A$ the set of all paths starting in i, ending in j, and passing through all other vertices of A in between. It will be convenient to fix the start and end points p and q of the paths, i.e., the search space is $X_{pq} := [p, X, q]$. With fixed p and q we need not treat the ends p and q as interface points, i.e., we can write $[A, j]$ for the path sets, where $p \in A$ and $q \notin A$ for all A. As for the TSP

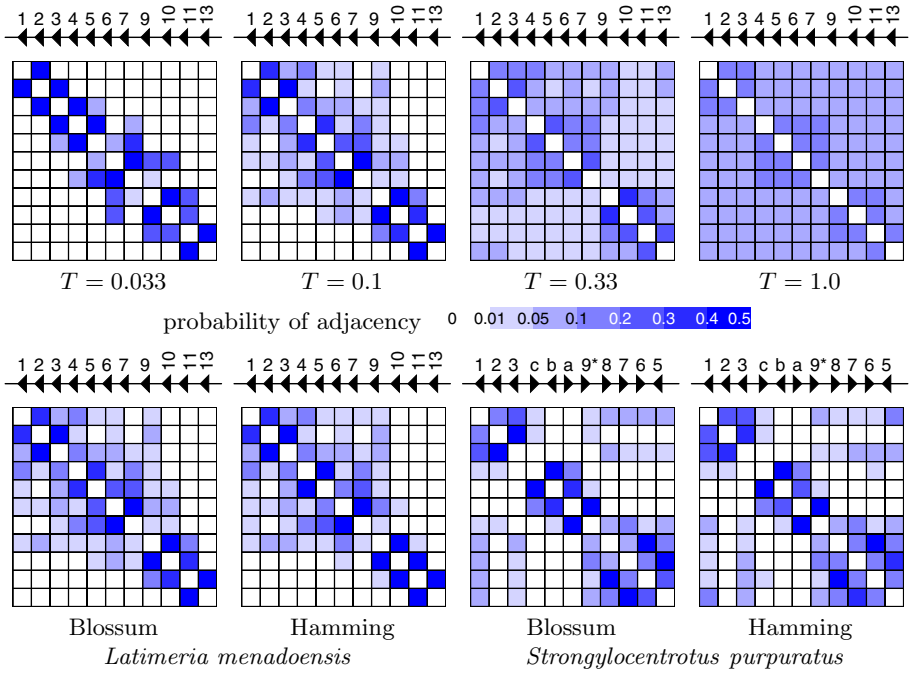


Fig. 1. Posterior probabilities of adjacencies of Hox genes along shortest Hamiltonian paths w.r.t. to phylogenetic distance. **Top:** effect of the temperature parameter T for distances between *Latimeria menadoensis* homeobox sequences. **Below:** Comparison of adjacencies for two different metrics (Hamming distance, and BLOSSUM-45 derived dissimilarities) in *L. menadoensis* (left) and *S. purpuratus*. $T = 0.1$ to emphasize the structure of the ambiguities. Note the adjacencies between the block of anterior Hox genes (1,2,3) and the middle group genes (5,6,7,8), reflecting the break-up and translocation of anterior genes to a genomic location before the posterior genes.

we have $[A, j] \mapsto [A \setminus \{j\}, k] +\!\!+ \langle k, j\rangle$ and $[A, j] +\!\!+ [A, j]^* = X_{pq}$ from which we obtain the outside objects as the path sets $[A, j]^* = [j, X \setminus A]$ with endpoint $q \in X \setminus A$. The corresponding concrete decompositions are $[j, B] \mapsto \langle j, k\rangle +\!\!+ [k, B \setminus \{j\}]$ for $k \in B \setminus \{j\}$. Partition functions Z over Hamiltonian paths are computed using $Z(A +\!\!+ B) = Z(A)Z(B)$, $Z([\{p\}, p]) = Z([q, \{q\}) = 1$, and $Z(\langle i, j\rangle) = \exp(-d_{ij}/kT)$ is the Boltzmann factor of the distance between two vertices, i.e., of the terminals. Our generalized ADP framework takes care of computing all $Z([p, A, i]) = Z([A, i])$ and $Z([k, B, q]) = Z([k, B])$. The *a posteriori* probability of observing an adjacency $i \sim j$ in path with fixed endpoints p and q is $P(i \sim j|p, q) = Z([p, A, i])Z(\langle i, j\rangle)Z([j, X \setminus (A \cup \{i\}), q])/Z(X_{pq})$.

As usual, this is simply the ratio of restricted and unrestricted partiton functions. Summing over the possible end points of the paths yields

$$P(i \sim j) = \frac{1}{Z} \sum_{p,q} Z([p, A, i])Z(\langle i, j\rangle)Z([j, X \setminus (A \cup \{i\}), q]) \,, \tag{6}$$

where $Z = \sum_{p,q} Z(X_{pq})$ is the partition function over all Hamiltonian paths. $Z(X_{p,q})/Z$ is the probability that the path has p and q as its endpoints.

Hox genes are ancient regulators originating from a single Hox gene in the metazoan ancestor. Over the course of animal evolution the Hox cluster gradually expanded to 14 genes in the vertebrate ancestor. Timing and positioning of Hox gene expression along the body axis of an embryo is co-linear with the genomic arrangement in most species. Only the 60 amino acids of the so-called homeodomain can be reliably compared at the extreme evolutionary distances involved in the evolution of the Hox system. We use either the Hamming distance, measuring the number of different amino-acids, or the transformation $d_{ab} = s(a,a) + s(b,b) - 2s(a,b)$ of the BLOSSUM45 similarity matrix to quantify the evolutionary distances of the homeodomain sequences. Setting k to the average pairwise genetic distance ensures that T quantifies the expected noise level as a fraction of the phylogenetic signal. For $T \to 0$ we focus on the (co)optimal paths only, while $T \to \infty$ leads to a uniform distribution of adjacencies.

We analyzed here the Hox A cluster of *Latimeria menadoensis* (famous as a particularly slowly evolving "living fossil"), which has sufferered the fewest gene losses among vertebrates. The Hox cluster of the sea urchin *Strongylocentrus purpuratus*, in contrast, has undergone fairly recent rearrangements of its gene order [13]. Fig. 1 shows the posterior probabilities of adjacencies. Both examples reflect the well-known clustering into anterior (Hox1-4), middle group genes (Hox4-8), and posterior ones (Hox9-13). The shortest Hamiltonian paths in *L. menadoensis* connect the Hox genes in their genomic order. In the sea urchin, however, we see adjacencies connecting the anterior subcluster (Hox1-3) with the genomic end of the cluster, i.e., the middle group genes (Hox8-Hox5).

4 Discussion

We have taken here the first step towards extending algebraic dynamic programming (ADP) beyond the realm of string-like data structures. Our focus is an efficient, yet notationally friendly way to treat DP on unordered sets. Our extension of ADP builds on the same foundation (namely `ADPfusion` [2]) as our grammar product formalism [14,4]. Our formalism explicitly redefines the rules of parsing to match the natural subdivisions of the data type in question. In the case of sets, these are bipartitions and the splitting of individual elements, rather than the subdivision of an interval or the removal of a boundary element that are at the heart of string grammars. As a showcase example we considered in detail the shortest Hamiltonian path problem, which arises e.g. in the context of the evolution of ancient gene clusters. In this context we are interested in particular in probabilities and hence in restricted partition functions. An ADP-style implementation and a principled approach to constructing outside algorithms is of particular practical relevance here.

Our current framework still lacks generality and completeness in several respects. The theoretical foundations for the automated calculation of outside grammars for, basically, traversals of arbitrary data types is our most immediate concern. In this context McBride's notion of a derivative operator acting on

data types [15] is highly relevant, even though it does not seem to be directly applicable. Even more generally, it might be possible to generate decomposition schemes, i.e. "grammar rules", from an analysis of the data structure itself.

Acknowledgements. This work was funded, in part, by the Austrian FWF, project "SFB F43 RNA regulation of the transcriptome", the Templeton Foundation, grant # 24332 "Origins and Evolution of Regulation in Biological Systems", and the DFG project "MI439/14-1".

References

1. Giegerich, R., Meyer, C.: Algebraic dynamic programming. In: Kirchner, H., Ringeissen, C. (eds.) AMAST 2002. LNCS, vol. 2422, pp. 349–364. Springer, Heidelberg (2002)
2. Höner zu Siederdissen, C.: Sneaking around concatMap: efficient combinators for dynamic programming. In: Proceedings of the 17th ACM SIGPLAN International Conference on Functional Programming, ICFP 2012, pp. 215–226. ACM (2012)
3. Sauthoff, G., Janssen, S., Giegerich, R.: Bellman's GAP - A Declarative Language for Dynamic Programming. In: Proceedings of the 13th international ACM SIGPLAN Symposium on Principles and Practices of Declarative Programming, PPDP 2011, pp. 29–40. ACM (2011)
4. Höner zu Siederdissen, C., Hofacker, I.L., Stadler, P.F.: Product Grammars for Alignment and Folding. IEEE/ACM Trans. Comp. Biol. Bioinf. 99 (2014)
5. Giegerich, R., Touzet, H.: Modeling Dynamic Programming Problems over Sequences and Trees with Inverse Coupled Rewrite Systems. Algorithms, 62–144 (2014)
6. Höner zu Siederdissen, C., Hofacker, I.L.: Discriminatory power of RNA family models. Bioinformatics 26(18), 453–459 (2010)
7. Voß, B., Giegerich, R., Rehmsmeier, M.: Complete probabilistic analysis of RNA shapes. BMC Biology 4(1), 5 (2006)
8. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. J. ACM 9, 61–63 (1962)
9. Held, M., Karp, R.M.: A dynamic programming approach to sequencing problems. J. SIAM 10, 196–201 (1962)
10. McCaskill, J.S.: The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers 29, 1105–1119 (1990)
11. Janssen, S.: Kisses, ambivalent models and more: Contributions to the analysis of RNA secondary structure. PhD thesis, Univ. Bielefeld (2014)
12. Elemento, O., Gascuel, O.: An efficient and accurate distance based algorithm to reconstruct tandem duplication trees. Bioinformatics 8(suppl. 2), S92–S99 (2002)
13. Cameron, R.A., Rowen, L., Nesbitt, R., Bloom, S., Rast, J.P., Berney, K., Arenas-Mena, C., Martinez, P., Lucas, S., Richardson, P.M., Davidson, E.H., Peterson, K.J., Hood, L.: Unusual gene order and organization of the sea urchin Hox cluster. J. Exp. Zoolog. B Mol. Dev. Evol. 306, 45–58 (2006)
14. Höner zu Siederdissen, C., Hofacker, I.L., Stadler, P.F.: How to multiply dynamic programming algorithms. In: Setubal, J.C., Almeida, N.F. (eds.) BSB 2013. LNCS, vol. 8213, pp. 82–93. Springer, Heidelberg (2013)
15. McBride, C.: Clowns to the left of me, jokers to the right (pearl): dissecting data structures. In: ACM SIGPLAN Notices, vol. 43, pp. 287–295. ACM (2008)

Using Binary Decision Diagrams (BDDs) for Memory Optimization in Basic Local Alignment Search Tool (BLAST)

Demian Oliveira, Fernando Braz, Bruno Ferreira,
Alessandra Faria-Campos, and Sérgio Campos

Department of Computer Science
Universidade Federal de Minas Gerais
Av. Antônio Carlos, 6627, Pampulha, 30123-970, Belo Horizonte, Brazil
demianbueno@yahoo.com.br,
{fbraz,bruno.ferreira,alessa,scampos}@dcc.ufmg.br

Abstract. Sequence alignment is the procedure of comparing two or more DNA or protein sequences in order to find similarities between them. One of the tools used for this purpose is the Basic Local Alignment Search Tool (BLAST). BLAST however, presents limits on the size of sequences that can be analyzed requiring the use of a lot of memory and time for long sequences. Therefore, improvements can be made to overcome these limitations. In this work we propose the use of the data structure Binary Decision Diagram (BDD) to represent alignments obtained through BLAST, which offers a compressed and efficient representation of the aligned sequences. We have developed a BDD-based version of BLAST, which omits any redundant information shared by the aligned sequences. We have observed a considerable improvement on memory usage, saving up to 63,95% memory, with a negligible performance degradation of only 3,10%. This approach could improve alignment methods, obtaining compact and efficient representations, which could allow the alignment of longer sequences, such as genome-wide human sequences, to be used in population and migration studies.

Keywords: Binary Decision Diagrams (BDD), Basic Local Alignment Search Tool (BLAST), Multiple Sequence Alignment.

1 Introduction

One of the fundamental problems in Bioinformatics is the alignment of sequences, which is used to compare and find similarities between primary biological sequences, such as DNA or proteins. Several algorithms have been proposed to address this issue. One of the most used is the Basic Local Alignment Search Tool (BLAST) [1]. However, it presents some limitations regarding the size of the sequences that can be analyzed. Therefore, improvements can be made to overcome these limitations.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 65–72, 2014.

In this paper, we present a different and novel approach to solve this problem, using the data structure Binary Decision Diagram (BDD), a special type of Binary Decision Tree (BDT), which allows a compressed, concise and efficient data representation. During the execution of BLAST, we represent the aligned sequences as a BDD, which discards redundant data shared by the sequences, which saves memory used by the program.

We have also performed a comparative study, measuring the execution time and memory usage of BLAST, available at the National Center for Bioinformatics Information (NCBI)[1], and our BDD-based version. We have observed a considerable improvement in memory usage, saving up to 63,95% memory, with a small trade-off of negligible performance degradation of only 3,10%.

This approach could improve alignment methods, obtaining compact and efficient representations, which could allow the alignment of longer sequences, such as genome-wide human sequences, to be used for population and migration studies.

2 Background

2.1 Basic Local Alignment Search Tool (BLAST)

Sequence alignment is the procedure of comparing two or more DNA or protein sequences by searching for series of individual characters that are in the same order in the sequence for the purpose of identifying similar regions, which may share characteristics, such as structure and function. Several algorithms exist to accomplish this, being prominent among them the **Basic Local Alignment Search Tool (BLAST)**. BLAST is used for the analysis, study and comparison of primary biological sequences, such as nucleotides in DNA and RNA sequences and amino acids in proteins [1,7]. A BLAST search enables a researcher to compare a query sequence with a library or database of sequences, and identify sequences that resemble it above a certain threshold. Different types of BLAST are available according to the sequences to be compared. A BLAST search allows the user to find alignments of a source biological sequence, called **query**, with another sequence, called **subject**, aiming to infer biological homology.

Since its creation, BLAST has been extended in different ways. Parallel implementations have been created, such as PLAST [8]. Specific optimizations of its algorithms, such as improving the order of its index seed, have also been performed [6]. BLAST+ , a complete reimplementation in C++ of BLAST (originally implemented in C), has also been created [3]. There are also non-equivalent alternatives for database search, such as BLAT [5], which uses the index of all nonoverlapping K-mers in the genome.

Finally, there are some works exploring the use of GPU and CUDA cores, for intensive parallel computations, such as GPU-BLAST [9] and G-BLASTN [10]. However, the use of a data structure known as Binary Decision Diagram (BDD) to improve the algorithm efficiency has yet not been pursued.

[1] NCBI website, `http://www.ncbi.nlm.nih.gov/`

2.2 Binary Decision Diagram (BDD)

The data structure **Binary Decision Diagram (BDD)** is derived from a Binary Decision Tree (BDT) – both are structures with two types of nodes, each node has two descendants. However, BDDs can be much more compact than BDTs because they discard redundant information, allow rapid graph traversal and check if two boolean functions are equivalent, as first described in [2]. BDDs are often used in symbolic model checking to represent finite states systems.

Although BDDs have several advantages, it has some drawbacks. The main one is the order of the variables (true or false) which appear in the boolean function being represented. Depending on it, the BDD can be heavily compressed or completely redundant. However, the problem of choosing the variable ordering which minimizes the BDD size is NP-complete [2].

Another issue with BDDs is the space complexity. Since BDDs are essentially an exhaustive representation of its boolean function, in the worst case it is exponential to the number of variables of the boolean function [2]. In practice, however, the worst case is uncommon, therefore this limitation often does not impact in the resulting BDD. Finally, there are several BDD extensions, such as Multi-terminal BDD (MTBDD), which can represent integer numbers as terminal nodes [4].

Given the main aspects of BDD and its ability to remove redundant information in a way that can change the algorithm behavior it may be used to improve BLAST implementation as proposed in this work.

3 Methods

3.1 BLAST

This work used **BLAST**, available at the National Center for Bioinformatics Information (NCBI), under the public domain license, as a source to perform the construction and representation of a BDD.

The work has focused on the program **Nucleotide Blast (blastn)**, responsible for the alignment of DNA nucleotide sequences. This is considered the most simple application to search for similarities between two biological sequences (query and subject), since a protein sequence represents the translation of a nucleotide sequence.

Some of the **blastn** functions have been modified to allow the construction of a BDD and its representation. We also added a method which allows measuring the performance of the system (in milliseconds) in order to find the impact of this approach on the algorithm.

For **blastn** the algorithm starts by dividing the data sequence to be searched (query) into words or K-mers. For a DNA sequence (nucleotides), the program has the default configuration of 11 nucleotides for the size of the word[2]. This value can be changed by the user, which can adjust how rigorous is the search.

[2] BLAST Help, `http://www.ncbi.nlm.nih.gov/books/NBK1763/`

After defining the words and creating the score matrix, which is used to rank similarities, the program has to search for a sequence in the **subject** which is equals to a word of the **query**. After obtaining a compatibility or match in the combination between the searched sequence (query) and the target database (subject), the algorithm shifts the word to a position with the objective of expanding the similarity and thus finding new matches. The algorithm also creates **gaps** between nucleotides in order to force similarities, which are represented by hyphens (-) in the output of results of a search.

3.2 BDD Construction

After the analysis of the **blastn** algorithm[3] for the alignemnts' representation, we have changed it to perform a pre-processing of the alignments with the creation of a BDD map, which has been used to visually represent the result of the alignment between a query and all subjects found by the initial processing of the algorithm. This BDD map was important to understand the result because it will probably add gaps in the sequences after finding a mismatch in their alignment, shifting the sequences to match their differences.

Algorithm 1. BDD-BLAST's algorithm. Clarification on the terms used: **query** is the sequence being aligned with several other sequences named **subjects**; the **tree** is the BDD being constructed; the **direction** is whether

```
1: for Each character of the query do
2:   if (Direction == Right) then
3:     for Each Comparison between query and subject do
4:       if (Comparison == true) then
5:         Create a node to the right of the tree with the value of the query
6:       else
7:         Create a node to the left of the tree with the value of the subject
8:       end if
9:     end for
10:  else
11:    for Each Comparison between query and subject do
12:      if (Comparison == true) then
13:        Create a node to the left of the tree with the value of the query
14:      else
15:        Create a node to the right of the tree with the value of the subject
16:      end if
17:    end for
18:  end if
19: end for
```

The second step was to create a model of the algorithm developed in this work. With the objective of building a balanced binary tree, our algorithm receives a

[3] `ncbi-blast-2.2.29+-src\c++\src\objtools\align_format\showalign.cpp`

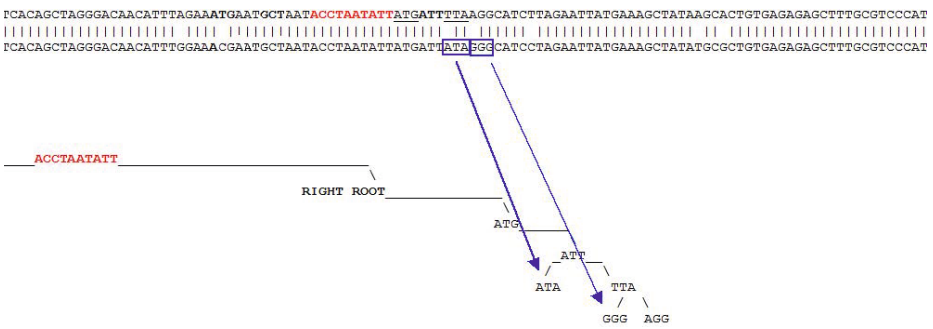


Fig. 1. BDD representation of a BLAST alignment. The central part of query (red) shows the tree root, where the BDD construction begins. The two indicated parts of the subject (blue) have not aligned with the query, therefore, both are added as innermost nodes.

representation of the alignments performed by BLAST (query versus subject) and, for each alignment found, executes the Algorithm 1.

The Algorithm 1 creates the BDD from its interior as mismatches of the alignment obtained in BLAST are found. The input are standard ACGT/U sequencesFigure 1 demonstrates a BDD produced by our algorithm for a word of size three (the pieces that the algorithm tries to match) and a root of size ten (the start of the alignment), where only last part of the alignment shows a discrepancy between query and subject.

Thus we can consider that the best case scenario for the construction of the BDD is the perfect alignment between query and subject, which will generate a binary tree that grows uniformly to both sides of its root. This means that the tree shows only matching nodes (left nodes on the left side and right nodes on right side). Mismatches makes the tree grow to its innermost nodes.

4 Results and Discussion

This Section presents our results and discussion: Section 4.1 describes the datasets and parameters used in our tests, Section 4.2 shows how we obtained our BDD representation and Section 4.3 presents memory usage results.

4.1 Parameters and Sequences

In order to demonstrate a real result in the execution of **blastn** through the site of the NCBI, a sequence registered by the code **NR_002605**, which represents the gene "Homo sapiens deleted in lymphocytic leukemia 1 (non-protein coding) (DLEU1), transcript variant 2, long non- coding RNA" will be used as reference.

The tests have been performed in an Intel Core i5 vPro 2.6 GHz with 4 GiB of RAM. The biological sequences used in the tests are NR_026084 *versus* **16S**

Microbial Sequences database and NM_000249, NM_00026, NM_15262 versus **NCBI Transcript Reference Sequences**. The searches have been executed using six different sample sizes: **Tiny** (79 nucleotides); **Small** (160 nucleotides); **Medium** (240 nucleotides); **Large** (560 nucleotides); **Full, real size** (1331 nucleotides); and **RNA human** (2662 nucleotides).

4.2 BDD Representation

The BDD-based algorithm proposed successfully represents the alignments obtained by BLAST as a BDD (Figure 1). Aligned sequences are represented as direct paths from the tree root, always to its right or left. On the other hand, parts of the sequence that have not aligned are shown in the tree in the innermost nodes. All the sequences can be recovered from the tree by traversing it, therefore no information is lost in this representation.

Outputting a binary tree in an organized way demands the insertion of several whitespaces between the nodes, which creates an exponential growth proportional to the size of the height of the tree. This has a direct impact on the size of the output file of the algorithm, which demands many hard disk writes, compromising the overall time taken by the algorithm execution.

We have compared the execution time of BLAST and the proposed BDD-based version. Results are shown in Table 1. The results are for the sizes of queries previously described. For this there are two configurations: BLAST (the original algorithm) and BDD (the proposed BDD-based version). The obtained execution time is the average of ten executions for each scenario, in order to avoid any transient fluctuations.

If the BDD was constructed without showing the BDD map (used for debugging purposes)and the resulting tree file output, a similar performance of the BDD-based version with the original algorithm is observed, as shown in Table 1. There is a small overhead between 0,75% and 3,10%, however, this is a small price paid for the huge benefit of memory optimization shown in the next section. Therefore, the construction of the BDD does not interfere significantly with the total execution time of BLAST.

Table 1. Comparison of the execution time of BLAST with the BDD-based version for different sizes of queries. The units are milliseconds (ms). The values are nearly identical, with a small overhead between 0,75% and 3,10%.

	Query Size					
Configurations	**Tiny**	**Small**	**Medium**	**Large**	**Full**	**RNA Human**
BLAST	29.63	60.10	124.21	398.02	902.09	1804.01
BDD	30.12	61.57	127.54	401.89	930.62	1860.76

4.3 Efficient Memory Usage

The memory usage of BLAST and the proposed BDD-based version has also been compared in this work (Table 2). The results are for the same query sizes and configurations. It has been observed a considerable improvement in memory usage, saving on average from 60,66% up to 63,95% memory. This result is a huge benefit with a small trade-off between performance and memory, since there is a negligible performance degradation (between only 0,75% and 3,10%).

These results have shown a scalability trend, since in all datasets the proposed BDD-based version saved 62,33% memory on average. Further studies using larger datasets are necessary to better corroborate the results and are in progress.

Table 2. Comparison of the memory usage of BLAST and the proposed BDD-based version for different sizes of queries. The average and median metrics are shown, and the units are bytes. The percent column (%) in each column group represents how much memory the BDD saved.

	Tiny			**Small**			**Medium**		
	BLAST	BDD	%	BLAST	BDD	%	BLAST	BDD	%
Average	206.59	74.72	63.95	462.60	176.47	61.94	389.53	145.34	62.79
Median	228.00	83.00	64.00	480.00	185.50	61.18	300.00	112.00	62.77
	Large			**Full**			**RNA Human**		
	BLAST	BDD	%	BLAST	BDD	%	BLAST	BDD	%
Average	1248.91	488.63	60.79	3633.01	1427.58	60.66	4434.55	1605.19	63.87
Median	1233.00	485.00	60.50	3549.00	1409.50	60.21	5772.00	2118.00	63.31

5 Conclusions and Future Work

This work has shown a novel approach to the representation of sequence alignments, which are used to compare and find similarities between biological sequences, such as DNA and protein. Our approach uses the data structure Binary Decision Diagram (BDD) to represent the results obtained by Basic Local Alignment Search Tool (BLAST). A BDD is an efficient and compact type of binary decision tree, removing any redundant information shared by the aligned sequences, which is often found in biological sequences. The proposed BDD-based version has shown a considerable improvement in memory usage, saving up to 63,95% memory, with a small trade-off of a negligible performance degradation of only 3,10%.

This approach can be used to improve current alignment methods, obtaining compact and efficient representations, which could allow the alignment of longer sequences, such as genome-wide sequences, to be used in population and migration studies.

Future work includes studying other scenarios and applications, using other BLAST types besides blastn for tests, comparing this approach with other sequence alignment methods besides BLAST, and modifying the algorithm to replace some of the BLAST functions.

References

1. Altschul, S.F., Gish, W., Miller, W., Myers, E.W., Lipman, D.J.: Basic local alignment search tool. J. Mol. Biol. 215(3), 403–410 (1990)
2. Bryant, R.E.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Comput. 35(8), 677–691 (1986)
3. Camacho, C., Coulouris, G., Avagyan, V., Ma, N., Papadopoulos, J., Bealer, K., Madden, T.L.: BLAST+: architecture and applications. BMC Bioinformatics 10, 421 (2009)
4. Clarke, E.M., Fujita, M., McGeer, P.C., McMillan, K., Yang, J.: Multi-terminal binary decision diagrams: An efficient data structure for matrix representation. In: IWLS 1993: International Workshop on Logic Synthesis, Lake Tahoe, CA, pp. 6a:1–6a:15 (1993)
5. Kent, W.J.: BLAT The BLAST-Like Alignment Tool. Genome Research 12(4), 656–664 (2002)
6. Lavenier, D.: Ordered index seed algorithm for intensive dna sequence comparison. In: IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–8 (2008)
7. Lipman, D.J., Pearson, W.R.: Rapid and sensitive protein similarity searches. Science 227(4693), 1435–1441 (1985)
8. Nguyen, V.H., Lavenier, D.: Plast: parallel local alignment search tool for database comparison. BMC Bioinformatics 10, 329 (2009)
9. Vouzis, P.D., Sahinidis, N.V.: Gpu-blast: using graphics processors to accelerate protein sequence alignment. Bioinformatics 27(2), 182–188 (2011)
10. Zhao, K., Chu, X.: G-blastn: accelerating nucleotide alignment by graphics processors. Bioinformatics (2014)

A Multi-Objective Evolutionary Algorithm for Improving Multiple Sequence Alignments

Wilson Soto[1,⋆] and David Becerra[2,⋆⋆]

[1] Intelligent Systems and Spatial Information Lab. Central University, Colombia
wsotof@ucentral.edu.co
[2] McGill Centre for Bioinformatics, McGill University, Montreal, Canada
david.becerraromero@mail.mcgill.ca

Abstract. Multiple Sequence Alignments are essential tools for many tasks performed in molecular biology. This paper proposes an efficient, scalable and effective multi-objective evolutionary algorithm to optimize pre-aligned sequences. This algorithm benefits from the great diversity of state-of-the-art algorithms and produces alignments that do not depend on specific sequence features. The proposed method is validated with a database of refined multiple sequence alignments and uses four standard metrics to compare the quality of the results.

1 Introduction

Multiple Sequence Alignment (MSA) refers to the computationally hard problem of aligning more than two sequences to identify evolutionary and/or structurally related positions. Over the past 20 years, novel computational methods that improve the accuracy of MSA tools have been proposed [1]. Currently, MSA tools are fairly reliable and are sufficiently fast to perform tasks of next generation sequencing [2], genome-annotation [3], structural and functional prediction [4], phylogenetic studies [5] and sequence data base searching [6].

State-of-the-art MSA algorithms cannot always guarantee consistent solutions, however, and there is no consensus on a strategy that produces optimal results [7]. Each algorithm has its own advantages and drawbacks when it faces particular sets of sequences, which can make it difficult for a given alignment problem to make a rational selection of an appropriate alignment tool. Some methodologies have been recently designed to combine distinct MSA algorithms to obtain extra consistency with a final alignment [8]. Furthermore, some optimization and computationally intelligent techniques have been applied to assemble sequences aligned by state-of-the-art algorithms predicting the expected accuracy of each alignment [9–11]. Finally, crowdsourcing platforms have also

⋆ Funding by Coimbra Group Scholarship Programme for Young Professors and Researchers from Latin American Universities. Acknowledges to Dr. Ion Petre and the Combio Laboratory at the Åbo Akademi University, Turku, Finland.
⋆⋆ DB is supported by Colciencias's Francisco Jose de Caldas scholarships.
The proposed algorithm is available at: http://201.245.199.28:8080/WebAlign/

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 73–82, 2014.

been used recently to improve the accuracy of pre-computed MSAs and to solve problems with local inaccuracies [12, 13].

Despite this relatively long history and a number of recent improvements, there is still a need for novel methodologies involving sequence alignments. In particular, there is an increasing need for faster MSA tools to deal with big data and for a way to overcome the limited performance of MSA tools on remote homologs and low similarity sequences. The dependency of MSA algorithms on particular sequence features are important bottlenecks in this area. Multi-objective optimization (MOOP) approaches are a methodology with several characteristics that are desirable for MSA optimization. Specifically, MOOPs can operate on a set of MSA candidate solutions, they are able to approximate solutions for optimization problems, and they can work on multiple conflicting objectives.

This paper proposes a fast, scalable and effective algorithm to optimize previously aligned sequences through a multi-objective approach. The algorithm is validated using a database of refined multiple sequence alignments (BAliBase) and uses four standard metrics to evaluate the quality of the predicted alignments.

2 Materials and Methods

2.1 MSA Algorithms

The proposed experimental framework has selected six state-of-the-art algorithms to provide seed alignments for optimization, namely: Clustal W [14], Clustal Omega [15], Muscle [16], MAFFT [17], ProbCons [18] and TCoffee [19]. It is important to stress that the proposed algorithm can be scaled with the insertion or deletion of any other MSA approach.

2.2 BAliBase

BAliBase [20] is a database of refined multiple sequence alignments commonly used to compare the performance of MSA programs. BAliBase performs validations on a wide spectrum of test cases in order to prevent the over-training of methods in a specific dataset. Each test case belongs to one of five possible reference sets [20]. (RV11 and RV12) is a set of equidistant PDB sequences in which any two sequences share $< 20\%$ identity. (RV20) are orphan sequences that all share $> 40\%$ identity. (RV30) are subfamilies where the sequences within a given subfamily share $< 40\%$ identity, but also where any two sequences from different subfamilies share $> 20\%$ identity. (RV40 and RV50) are sequences that share $> 20\%$ identity with at least one other sequence, but that include sequences that contain large N/C-terminal extensions or internal insertions respectively BAliBASE maintains the high quality of alignments through the use of 3D structural superpositions combined with a final manual validation and refinement step. The number of sequences in version 3.0 was increased from 1444 to 6255, composed of 218 alignments.

2.3 Multi-Objective Evolutionary Algorithm (MOEA)

MOEAs belong to the class of stochastic optimization algorithms that mimic natural evolution to solve a multi-objective optimization problem (MOOP). A MOOP problem can be defined as the problem of finding a vector $x = [x_1, x_2, \ldots, x_n]^T$ such that:

i) satisfies the r equality constraints $h_i(x) = 0,\ 1 \leq i \leq r$,
ii) is subject to the s inequality constraints $g_i(x) \geq 0,\ 1 \leq i \leq s$
iii) optimizes the vector function $f(x) = [f_1(x), \ldots, f_m(x)]^T$.

The vector x is an n-dimensional decision vector and $\mathcal{X}$ is the decision space, *i.e.*, the set of all expressible solutions. The objective vector $f(x)$ maps $\mathcal{X}$ into $\Re^m$, where $m \geq 2$ is the number of objectives. The image of $\mathcal{X}$ in the objective space, is the set of all attainable points.

The concept of optimality is introduced through the notion of Pareto optimality. A vector $x \in S$ is said to be Pareto optimal if all other vectors $x^* \in S$ have a higher value for at least one of the objective functions in $f(x)$. A Pareto front is the image of all solutions. The points that form the shape of the Pareto front are called non-dominated points. The solutions in the final set are expected to be close to optimal and non-dominated with respect to one another.

MOEAs are algorithms that simulate natural evolution by an iterative computation process in which a set of candidate solutions are subsequently modified and improved through selection and variation procedures until some level of acceptable quality is met. The algorithm takes a given MSA (i.e., original population) and returns an improved one after performing a series of mutations and crossover operations. In other words, the procedure "evolves" the original MSAs to produce a better one. Generally, a MOEA schema can be divided into the following phases: generation, evaluation, and selection of individual (See fig 1).

In a MOEA, individuals contain the information of solutions (i.e., alignments). Alignments are represented as a $n \times m$ multi-array of integers, where n is the number of sequences and m is the length of the alignment. If a cell in the multi-array corresponds to a match, then it contains the position of the amino-acid in the protein sequence. On the other hand, if a cell corresponds to a gap, then it contains the position in the protein sequence of the previous match (See the multi-array depicted in the boxes Crossover or Mutation in figure 1). This representation was chosen given its suitability in the implementation of genetic operators [9].

This paper constructs the initial population (i.e. set of individuals) using the alignments produced by state-of-the-art MSA algorithms. This population is composed of the alignments reported for each algorithm plus some genetically modified versions (i.e. versions that apply genetic operators) of those alignments. Subsequent populations are then improved using a genetic algorithm. Genetic operators (crossover and mutation) then produce new generations through the selection of new individuals based on a fitness function. Finally, a stopping criterion is achieved when for a fixed number of iterations no improvements have been made.

The mutation and crossover operators represent mechanisms to perform exploration (increasing the diversity of a population) and exploitation (increasing the depth of the search) procedures within the search space, respectively.

An implemented mutation randomly selects the positions of one gap and one amino-acid in an individual. It then introduces the gap after the amino-acid position and performs a shift of one position for all the positions between the selected amino-acid and the end of the sequence or the position of the selected gap, whichever occurs first (See figure 1).

A two-point crossover, where each point represents a column in the alignment, is then implemented. Once two points have been randomly selected in one individual (parent one), the positions between the two points, which correspond to the same string of position indexes, are then sought in another random individual (parent two). Finally, the strings of positions are exchanged between the two parents (See figure 1).

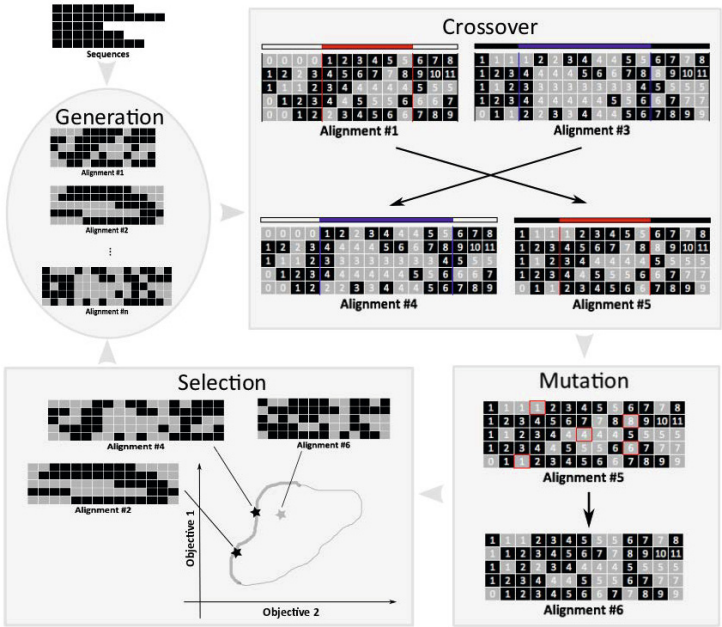


Fig. 1. The proposed algorithm

Evaluating Individuals

In this proposed algorithm, the evaluation of individuals is achieved through a fitness function that considers two objective functions, entropy and the metric MetAl [21]. Entropy measures the variability of an MSA by defining the frequencies of the occurrence of each letter in each column. Entropy is minimal when

the same symbol is always present in one position, while it is maximal when all symbols are present with the same frequency. The total entropy for an MSA is the sum of the entropies of its columns (see equation 1), where P_i is the fraction of residual amino acid type i, M is the number of amino acid types plus the gap character, and N is the number of columns in the MSA. For the purposes of this algorithm, good alignments are considered to be those that minimize their total entropy.

$$H = \sum_{j=1}^{N} - \sum_{i=1}^{M} P_i \log_2 P_i \tag{1}$$

Frequency-based approaches, as a measure of entropy, do not consider the positional and evolutionary characteristics of residue presented in an MSA. The approach proposed here uses a combination of four metrics that incorporate positional and evolutionary information, which are processed by MetAl software to compute a single score between a target (the offspring) and a set of sequences (the parents). This score is based on: (i) a simple correction to the SP score; (ii) raw gap information; (iii) positions of gaps occurring in a sequence; and (iv) positions of indel events occurring in a sequence and on its phylogenetic tree This single score is then used as a second objective, one that is to be minimized.

Selecting Individuals

The selection process drives searches towards regions containing the best individuals to compute the Pareto front. In order to generate the Pareto front for each generation, each population of a generation is sorted according to a non-dominated sorting approach. A population is sorted into different non-domination levels, and the Pareto front is filled with individuals belonging to the best ranked levels until the desired number of individuals for a population is reached. Since MOEAs require a phase of high level of information to report one solution from the Pareto front, an algorithm was used based on the identification of the knees [22], i.e. the regions in the Pareto front where small displacements produce a big detriment to at least one of the objectives.

Validation

We validate our predictions using the Sum of Pairs (SP), Total Column (TC), MetAl and hypervolumen metrics based on the BAliBase benchmark.

The SP and TC metrics were computed by the *BaliScore* script. Equations 2 and 3 define these metrics, where M_r is the number of columns in the reference alignment and S_{ri} is the score S_i for the i-th column in the reference alignment, $p_{i,j,k} = 1$ if the pair of residues $A_{i,j}$ and $A_{i,k}$ are aligned with each other in the reference alignment, and $p_{i,j,k} = 0$ otherwise. On the other hand, $C_i = 1$ if all the residues in the i-th column are aligned in the reference alignment. Otherwise, $C_i = 0$. The possible values for SP and TC range from [0,1], and a score equal to one represents an exact agreement between the alignments.

$$SP = \frac{\sum_{i=1}^{M} S_i}{\sum_{i=1}^{M_r} S_{ri}}, \quad where \quad S_i = \sum_{j=1}^{N} \sum_{k=1, k \neq j}^{N} p_{i,j,k} \tag{2}$$

$$TC = \sum_{i=1}^{M} \frac{C_i}{M} \tag{3}$$

SP and TC are standard scores for comparing the performance of MSAs. However, many concerns have been raised about their use. The MetAl score [21] can be used as a metric to fix inaccuracies and to incorporate additional information in the evaluation process. The MetAl metric incorporates a correction to the SP score as well as gap and indel information when computing the score between a target and a set of sequences. The range for the MetAl score is [0,1], where a perfect match is found when the score is equal to 0 (plotted as $1 - MetAl$ in figure 2). It is important to stress that the MetAl score uses the BAliBase reference alignments as its target in the validation process, in contrast to the evaluation process, where the parents are used as targets. As such, the BAliBase alignments are strictly used only during the validation process to guarantee a blind and fair comparison between the MSA tools.

The hypervolume is a metric used by researches to measure the quality of a Pareto front. The hypervolume indicator measures the volume of the dominated portion of the objective space. The hypervolume represents in a unary value the spread of solutions along the Pareto front, as well as the distance of a set of solutions from the Pareto-optimal front. The hypervolume has two highly desirable features: it is sensitive to any improvements, and it guarantees that any approximation set that achieves the maximally possible quality value for a particular problem contains all Pareto-optimal objective vectors [23]. The hypervolume indicator I_H for a solution set $A \subset \Re^d$ can be defined on the basis of a reference set $R \subset \Re^2$, as shown in Equation 4. In that equation, the symbol λ stands for the Lebesgue measure and the set $H(A, R)$ denotes the set of objective vectors that are enclosed by the front $F(A)$ given by A and the reference point R.

$$I_H(A) := \lambda(H(A, R)) \tag{4}$$

3 Results

The MOEA algorithm was executed based on the following parameters: 7 independent runs over 218 alignments, with each population containing 56 individuals. The crossover and mutation probabilities were set to 0.3 and 0.1, respectively, and a stopping criterion was deemed to be achieved at 5 consecutive generations with no improvements in the hypervolume.

The input of our algorithm is a set of pre-aligned sequences and the output is the prediction of a single alignment belonging to the non-dominated set. This output is compared with the results given by the six state-of-the-art algorithms, taking as target the BAliBase benchmark and four validation scores. For the

TC score, our approach achieved optimum results in 5 out of the 6 groups, and second best results in the remaining group (RV50). For the MeTal score, it achieved optimum results in 3 out of the 6 groups (RV11, RV12, RV30), and second best results in the three remaining groups (RV20, RV40, RV50). With respect to the SP score, the proposed approach is in the top 2 for 4 out of 6 groups.

Figure 2 shows the proportion of alignments with the best scores reported for each MSA tool. The proposed algorithm demonstrates excellent performance for all six groups. It performed outstandingly for groups RV11 and RV12, where the proposed algorithm outperformed the other MSA tools for all three scores. The excellent performance of the proposed algorithm is more easily observed in the MetAl and TC scores than in the SP score. For example, it found 60% and 48% of the best MeTal and TC scores in the RV12 and RV11 sets, respectively. It is important to note that the proposed approach ranked in the top 2 for all groups based on the MeTal and TC scores. The proposed approach predicted the best alignments for two groups (RV11, RV12), the second best in two groups (RV30, RV50) and the third best in the remaining groups (RV20, RV40).

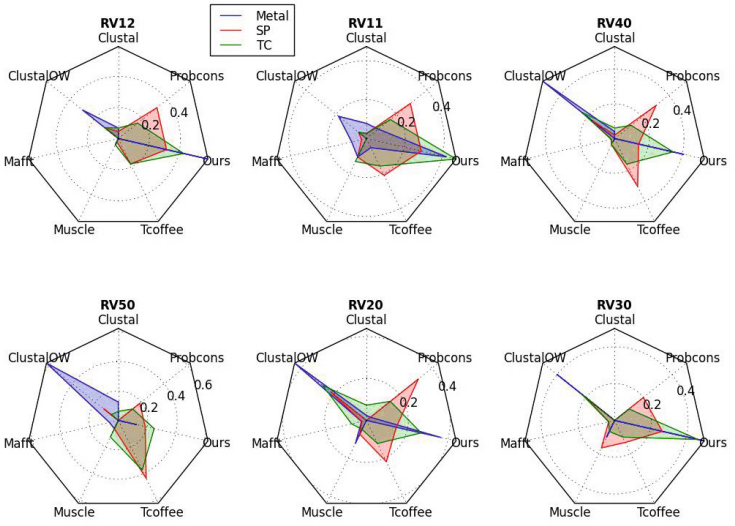


Fig. 2. Proportion of alignments with the best scores reported for each MSA tool. The results are clustered in 6 different hierarchical groups as defined by BAliBase; each vertex in the polygon represents an MSA tool and an area is plotted with regards the proportion of best alignments reported for the SP, TC and (1-MetAl) scores.

Figure 3 reports the average error obtained for each MSA tool. This error is defined by equation 5, where $S \in \{MetAl, SP, TC\}$. The function $score(i)$ returns the score of the alignment i generated by the metric S, and the function $best(i)$ returns the best score for the alignment i using the metric S on all the

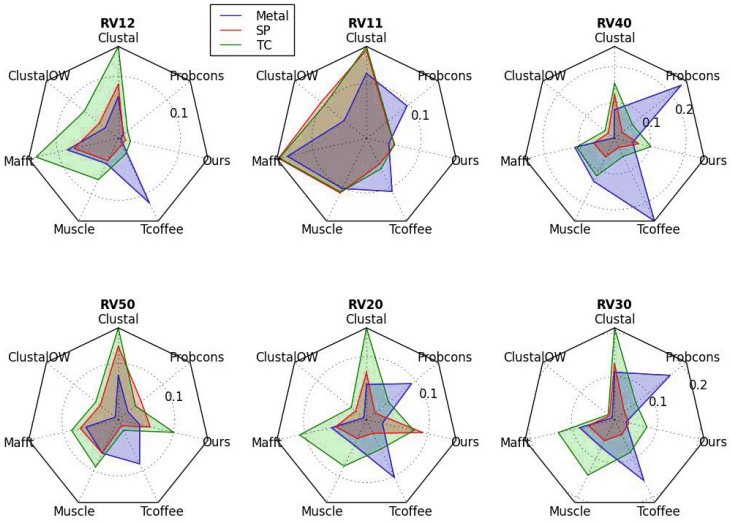


Fig. 3. Average error obtained for each MSA tool. The results are clustered in 6 different hierarchical groups as defined by BAliBase; each vertex in the polygon represents a MSA tool and an area is plotted with regards the average error (See equation 5) of the scores obtained by the MSA tools.

MSA tools. From the figure, one can observe that the proposed approach works very consistently with all groups for each of the three score schemas in the six different groups. Its degree of error is especially low in the RV11, and RV12 groups, where for each of the three score schemes it obtained the best error compared to the other MSA tools. This conclusion concurs with that reached by an analysis of the same sets in figure 2. The ClustalOW algorithm behaved similarly in the groups RV30 and RV40, however it had a higher number of errors for the RV11 and RV12 sets.

$$error_S = \sum_{i=1}^{M} |best_S(i) - score(i)| \tag{5}$$

Figure 4 reports the quartiles of the gained hypervolume through the computation of the generations in the MOEA. The gain of a specific alignment is computed as the difference between the hypervolume of the last and first Pareto fronts. In figure 4, it is clear that the volume of the dominated portion of the objective space increased through the generations (i.e, values greater than zero). The proposed algorithm was able to then push a set of already aligned sequences (the initial population) towards the Pareto optimal solutions (increasing the hypervolume in 204 (94%) alignments). The worst performance is found in the RV11set, where seven alignments did not improve their hypervolumes. On the other hand, the RV50 set improved the hypervolume of all its alignments.

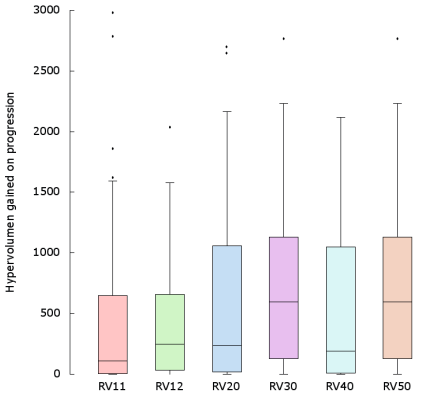


Fig. 4. Boxplots of the hypervolume gained on progression of the MOEA algorithm

4 Conclusions

This work contributes to the solution of the MSA problem by proposing a novel algorithm to optimize previously aligned sequences. The proposed model is based on a MOEA, which, as this paper demonstrated, provides an adequate exploration of the search space. Moreover, the proposed strategy improves the accuracy of the MSAs used as inputs for the model. The proposed algorithm is not the holy grail of MSA tools, and does not propose to be a method that will outperform all the other MSA tools on any possible sequence. However, it is a method that, with a very reasonable cost in CPU time, produces more accurate alignments than the alignments obtained from other methodologies. Furthermore, it proved to be less dependent on specific features of sequences and very stable and robust when used on diverse biologically targeted sequences.

The proposed approach is more than just a static algorithm. It is also a pipeline that allows for the optimization of different MSA tools. In this paper, six different MSA tools were chosen to develop a study case, however, these can be replaced with others. This feature is important because it will allow our algorithm to be tested and used over a diverse range of representative situations, and it will be able to accommodate new MSA tools to its pipeline as they become available.

References

1. Wallace, I.M., Blackshields, G., Higgins, D.G.: Multiple sequence alignments. Current Opinion in Structural Biology 15(3), 261–266 (2005)
2. Li, H., Homer, N.: A survey of sequence alignment algorithms for next-generation sequencing. Briefings in Bioinformatics 11(5), 473–483 (2010)
3. Blanchette, M.: Computation and analysis of genomic multi-sequence alignments. Annu. Rev. Genomics Hum. Genet. 8, 193–213 (2007)
4. Marks, D.S., Hopf, T.A., Sander, C.: Protein structure prediction from sequence variation. Nature Biotechnology 30(11), 1072–1080 (2012)

5. Wang, L., Leebens-Mack, J., et al.: The impact of multiple protein sequence alignment on phylogenetic estimation. IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB) 8(4), 1108–1119 (2011)
6. Altschul, S.F., Madden, T.L., et al.: Gapped blast and psi-blast: a new generation of protein database search programs. Nucleic Acids Research 25(17), 3389–3402 (1997)
7. Nuin, P., Wang, Z., Tillier, E.: The accuracy of several multiple sequence alignment programs for proteins. BMC Bioinformatics 7(1), 471 (2006)
8. Wallace, I.M., O'Sullivan, O., et al.: M-coffee: combining multiple sequence alignment methods with t-coffee. Nucleic Acids Research 34(6), 1692–1699 (2006)
9. Ortuño, F., Florido, J.P., et al.: Optimization of multiple sequence alignment methodologies using a multiobjective evolutionary algorithm based on nsga-ii. In: 2012 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2012)
10. Ortuño, F., Valenzuela, O., et al.: Predicting the accuracy of multiple sequence alignment algorithms by using computational intelligent techniques. Nucleic Acids Research 41(1), e26–e26 (2013)
11. Ortuño, F., et al.: Optimizing multiple sequence alignments using a genetic algorithm based on three objectives: structural information, non-gaps percentage and totally conserved columns. Bioinformatics (29), 2112–2121 (2013)
12. Kawrykow, A., Roumanis, G., et al.: Phylo: a citizen science approach for improving multiple sequence alignment. PloS One 7(3), e31362 (2012)
13. Kwak, D., Kam, A., et al.: Open-phylo: a customizable crowd-computing platform for multiple sequence alignment. Genome Biology 14(10), R116 (2013)
14. Thompson, J.D., et al.: Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Research 22(22), 4673–4680 (1994)
15. Sievers, F., Wilm, A., et al.: Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. Molecular Systems Biology 7(1) (2011)
16. Edgar, R.C.: Muscle: multiple sequence alignment with high accuracy and high throughput. Nucleic Acids Research 32(5), 1792–1797 (2004)
17. Katoh, K., et al.: Mafft: a novel method for rapid multiple sequence alignment based on fast fourier transform. Nucleic Acids Research 30(14), 3059–3066 (2002)
18. Do, C.B., Mahabhashyam, M.S., et al.: Probcons: Probabilistic consistency-based multiple sequence alignment. Genome Research 15(2), 330–340 (2005)
19. Notredame, C., et al.: T-coffee: A novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology 302(1), 205–217 (2000)
20. Thompson, J.D., et al.: Balibase 3.0: latest developments of the multiple sequence alignment benchmark. Proteins: Structure, Function, and Bioinformatics 61(1), 127–136 (2005)
21. Blackburne, B.P., Whelan, S.: Measuring the distance between multiple sequence alignments. Bioinformatics 28(4), 495–502 (2012)
22. Branke, J., Deb, K., Dierolf, H., Osswald, M.: Finding knees in multi-objective optimization. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 722–731. Springer, Heidelberg (2004)
23. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 862–876. Springer, Heidelberg (2007)

BION2SEL: An Ontology-Based Approach for the Selection of Molecular Biology Databases

Daniel Lichtnow[1], Ronnie Alves[2,5,6], Oscar Pastor[3], Verónica Burriel[3], and José Palazzo Moreira de Oliveira[4]

[1] Universidade Federal de Santa Maria, UFSM, Santa Maria, RS, Brazil
dlichtnow@politecnico.ufsm.br
[2] Institut de Biologie Computationnelle, Montpellier, France
alvesrco@gmail.com
[3] Universitat Politècnica de València, Valencia, Spain
{vburriel,opastor}@pros.upv.es
[4] Universidade Federal do Rio Grande do Sul, UFRGS, RS, Porto Alegre, Brazil
palazzo@inf.ufrgs.br
[5] Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier, UMR 5506, Université Montpellier 2, Centre National de la Recherche Scientifique, Montpellier, France
[6] PPGCC, Universidade Federal do Pará, Belém, Brazil

Abstract. The catalogs of molecular biology databases does not provide a full description of databases, so the user should select databases using limited information available. Taking into account this fact, in the context of an initiative called *BioDBCore*, a group of experts proposes core metadata definitions to describe the molecular biology databases. However, how to use these metadata to infer the quality of a database is a clear open issue. In the present work, we propose an ontology-based approach aiming to guide the database selection process from molecular biology database catalogs using these metadata.

Keywords: molecular biology databases, database catalog, data quality, ontology, database selection.

1 Introduction

There have been some initiatives to create online molecular biology database collections. Currently the database selection processes from these catalogs is an iterative process, where the users must evaluate manually a set of candidate databases using limited query mechanisms and incomplete information about databases. In the present work, we propose an ontology-based approach to the selection of database catalogs. We start showing examples of molecular biology database catalogs (Section 2). After, an overall perspective of an approach that can help a user in the selection process is presented (Section 3). A preliminary version of the proposed ontology and its data quality rules are presented (Section 4) and an application scenario illustrating the use of the ontology is discussed (Section 5). Final remarks and future works conclude this paper (Section 6).

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 83–90, 2014.

2 Related Work

The *Nucleic Acids Research Database Collection* is one of the most well-known molecular biology database catalog. This catalog is published since 1996 and its size has been increasing constantly (in 2001, this catalog included only 96 databases and in 2014 it includes 1,552 databases [1]). In general, the molecular biology database catalogs offer limited information regarding the databases and the querying capabilities. It is not possible for a user to indicate any additional requirement and there is not any information about the quality of the databases (e.g. reputation, accuracy, etc.). Thus, the database selection processes from these database catalogs use to be an iterative process where, generally, users must carefully evaluate the set of candidate databases using only categories which have been already provided by these catalogs. For these reasons, there are several initiatives to create richer database catalogs and the *BioDBCore* consortium proposed a set of metadata elements for the better description of molecular biology databases [2]. Despite these initiatives, to the best of our knowledge, a little investigation has addressed the use of the database metadata (specially *BioDBCore* attributes) on the identification of the best databases.

3 The Proposed Approach

The background of our approach is an ontology for the evaluation of the overall quality of a molecular biology database. In the approach, a user indicates some quality requirements and databases of a catalog are evaluated using a set of rules defined in the ontology. The result of the evaluation consists of a set of databases classified according to distinct quality levels (*High*, *Medium*, *Low*) of each data quality dimension (e.g. accuracy, believability, etc.) considered into the evaluation. These three levels are similar to those used in *CASIMIR Database Description Framework* [3]. In [3] users must manually assign each database to a quality level, our proposal is to assign each database to a quality level using database metadata and ontology rules.

Thus, the overall evaluation of a database is carried out using metadata defined in the *BioDBCore* initiative that is represented in the proposed ontology. The utilization of the metadata in the data quality evaluation process has been considered in some works where metadata models are matched to data quality model requirements [4]. In order to identify potential quality dimensions to the proposed approach, we started by exploring quality dimensions defined by Wang and Strong, due to the relevance of this work [5] in the data quality community. Next, we adapted these quality dimensions following a process similar to Naumann et al. [6] who selected the quality dimensions taking into account the data integration process of molecular biology information systems. In the quality dimensions selection process the metadata defined in *BioDBCore*, quality dimensions, and quality indicators evaluated on previous works [7,8] are also considered.

Table 1. *BioDBCore* metadata elements and quality dimensions

Element	**Quality Dimension**
Database name	None
Main resource URL	Believability
Contact information (e-mail; postal mail)	Ease of understanding
Date resource established (year)	Timeliness
Conditions of use (free, or type of license)	License
Scope: Data types	Relevancy
Scope: Curation policy	Accuracy
Scope: Standards Used	Verifiability Completeness
Standards: MIs	Verifiability Completeness
Standards: Data formats	Interpretability
Standards: Terminologies	Representational Consistency
Taxonomic coverage	Relevancy
Data accessibility/output options	Accessibility
Data release frequency	Timeliness
Versioning period and access to historical files	Timeliness
Documentation available	Ease of understanding
User support options	Ease of understanding
Data submission policy	Accuracy
Relevant publications	Believability
Resource's Wikipedia URL	Ease of understanding
Tools available	Accessibility

The Table 1 shows the quality dimensions defined and the metadata element proposed by *BioDBCore*. Because of the lack of space, we do not present, quality dimension definitions (see [5]). In Table 1, the metadata element *Scope: Data type* refers to the content of a database (e.g. a database can store microarray experiments, sequence references, protein structure, etc). The metadata elements *Standards: MIs (Minimum Information)*, *Standards: Data formats* and *Standards: Terminologies* refer to standards present in a catalog of standards - *BioSharing*. One example of a standard is *MIAME - Minimum Information About a Microarray Experiment* - whose main goal is to specify all the data necessary to interpret a microarray experiment. Some quality indicators can be derived from some metadata elements in Table 1 (e.g. the element *Main resource URL* can be used to obtain the *PageRank* of the databases homepage and the number of web links pointing to the database's homepage).

Some of these metadata elements were tested in the evaluation process of molecular biology databases in a previous work [7]. In this previous work, a comparison was done between the rankings generated using these indicators and rankings which had been manually generated by three experts using Spearman correlation (all correlations are significant for $p < 0.05$). However, the results also indicate that it is important to take into account aspects related to more specific

users' needs. In this sense, the database categories present in the *Nucleic Acid Research* catalog are not enough for guiding the selection process of databases. This particular observation points out an interesting opportunity for developing an ontology-based approach for the selection problem.

4 An Ontology-Based Approach to Guide the Selection of the Best Molecular Biology Databases

Taking into account the quality dimensions, the metadata elements proposed in *BioDBCore* and our previous works, a prototype of an ontology-based approach to guide the selection of the best molecular biology databases, called in short *BION2SEL*, has been defined. The *BION2SEL* is logically separated into a set of distinct components (ontologies). The two main components are (1) The *Molecular Biology Database* ontology where database features are described using the metadata elements proposed in *BioDBCore* and (2) The *Molecular Biology Database Quality* ontology where the quality dimensions are defined.

The *Molecular Biology Database* ontology describes the properties of molecular biology databases. Each instance of this ontology is a molecular biology database. The properties are basically derived from the metadata elements present in *BioDBCore* (see Table 1). Some values assigned to the properties of this ontology are vocabularies defined in other ontologies. Thus, there are ontologies that provide vocabularies (e.g. population class, data format). For example, the range of *Molecular Biology Database* ontology property *data types* refers to classes of *Molecular Biology Summary Data*. The *Molecular Biology Summary Data* ontology was created by us and can be considered as a summarized global schema of all databases present in a database catalog. In the *Molecular Biology Summary Data* ontology, the relationships between classes are also defined (e.g. *Protein hasStructure ProteinConformation*). Using the *Molecular Biology Summary Data* ontology classes, the *OMIM* database is assigned to classes *Gene*, *Allele*, *Allele Variant* and *Disease* classes and the *ALFRED* database is assigned to classes *Gene*, *Allele*, *Allele Variant* and *Population*. Thus, the user should indicate the biological entities (e.g. gene, pathways, etc.) defined in the ontology (this process is similar to adopted in [9]).

The *Molecular Biology Database Quality* ontology defines the quality criteria to evaluate a molecular biology database according to distinct quality dimensions. For each quality dimension, three quality levels are defined in the ontology: *High*, *Medium* and *Low*. A database, an instance of *Molecular Biology Database*, is classified into a quality level in accordance to the quality indicators corresponding to metadata elements (properties). The conditions to classify a database into a specific quality level are defined through *SWRL - Semantic Web Rule Language (SWRL)*, a rule language based on OWL and *SQWRL - Semantic Query-Enhanced Web Rule Language* [10]. Next, examples of rules defined for some quality dimensions (relevancy, believability and accessibility) are shown.

The relevancy is the quality dimension which has been initially considered in the database selection process. As the goal is to select databases from a catalog

without access its content, we define relevancy as the percentage of classes of objects (e.g. gene, protein, etc.) required by the user rather than the information stored on a database. Thus, in the proposed ontology, the databases with all required data are classified as having a *High_Relevancy*, databases with 50% or more of required data are classified as a *Medium_Relevancy*, and databases with less than 50% of required data are considered having *Low_Relevancy*. In (1) a rule defined with *SWRL* and *SQWRL* to classify a database as a *High_Relevancy* database is shown.

Basically, the rule defines that a database - *Molecular_Biology_Database(?x)* - must have required data. The metadata element *data_types* contains the names of the classes of *Molecular Biology Summary Data* ontology present in a database. Thus using *SQWRL* two sets are compared: one set contains the classes of *Molecular Biology Summary Data* required by a user *(?ds)* and another set contains the classes of *Molecular Biology Summary Data* that a database has *(?s)*.

$$\begin{aligned} &Molecular\_Biology\_Database(?x) \wedge \\ &data\_types(?x, ?dt) \wedge \\ &sqwrl : makeSet(?s, ?dt) \wedge \\ &sqwrl : groupBy(?s, ?x) \wedge \\ &sqwrl : makeSet(?ds, Gene) \wedge \\ &sqwrl : makeSet(?ds, Disease) \wedge \\ &sqwrl : difference(?dif, ?ds, ?s) \wedge \\ &sqwrl : size(?sdif, ?dif) \wedge \\ &swrlb : equal(?sdif, 0) \rightarrow sqwrl : select(?x) \end{aligned} \tag{1}$$

Believability is related to the content creator and the explicit or implicit users' ratings [4]. Implicit users' ratings can be the number of references in the web to a particular database. Some previous experiments have demonstrated that the number of paper citations related to databases is an interesting quality indicator [8]. Thus, rules based on the number of citations of these papers to classify a database as a database with high, medium, and low believability are defined. At the present moment, the defined rules take into account the median of the number of citations. Thus, databases where papers are cited higher than the median are considered databases with *High_Believability*, databases where the number of citations is equal to the median are considered databases with *Medium_Believability*, and databases where the number of citations is lower than the median are considered databases with *Low_Believability*.

Regarding to accessibility, one possibility is to measure the accessibility degree of a database considering all access mechanisms available. Thus, we defined that databases are considered to have *High_Accessibility*, if they allow to inform an specific argument for a query and returns an specific data.

5 Ranking Candidate Gene Markers in Alzheimer: An Ontology-Based Quality Model Experience

In this section, some aspects of the utilization of the *BION2SEL* to guide researchers who need to retrieve data from a set of molecular biology databases

are illustrated. The quality dimensions considered here are accessibility and relevancy. Believability is not considered because, in the example, we are considering a small number of databases (if two or more databases have the same relevancy degree, databases with higher believability must be selected first). The case of use consists of identifying the most differentially expressed genes (*DEG*) in brains affected by Alzheimer's disease (documented in [11]). In the present case of use, the *DEG* task in brains affected by Alzheimer's disease can be subdivided in a set of subtasks:

1. Obtaining transcriptomic (*Microarrays*) data related to Alzheimer;
   (a) Selecting a set of transcriptomic studies;
   (b) Selecting the microarrays experiments related to hippocampus;
2. Conducting the differential expression analysis i.e. identifying *DEG*;
   (a) Ranking probes according to a two-group (normal *vs.* affected tissue)
   (b) Mapping probes to genes. This is an annotation process (genes names are assigned to probes);
3. Exploring functional enrichment analysis using the *DEG* set;
4. Identifying and discarding genes whose codified proteins are not known;
5. Verifying whether the selected candidate gene set is related to the Alzheimer.

The database selection process starts by exploring aspects related to relevancy. After other quality dimensions can be evaluated (e.g. believability). Thus, for each task, when it is necessary, the user indicates the required data, i.e. the biological entities of *Molecular Biology Summary Data*. Table 2 presents a set of databases with a list of biological entities of *Molecular Biology Summary Data* present in each database (this list is not exhaustive).

The requirement related to task 1 can be expressed by a query like "retrieve all gene expression studies related to Alzheimer". Thus, this query must be executed in a database which contains data about gene expression studies and diseases. In this sense, the relevant classes in the *Molecular Biology Summary Data* are *Assay* and *Disease*. Thus, the most relevant database is *ArrayExpress Archive*. The *ArrayExpress Archive* has an access mechanism (an *HTML form*) which allows to retrieve gene expression experiments using as argument the name of a disease.

The requirement related to task 3 can be expressed by a query like "returns the function of a set of genes". Thus, the query must be executed in a database that contains data about gene function. The most appropriate database is *Gene Ontology*. The next step verifies which identified genes code to known proteins (task 4). In this case, the user need can be expressed by "identifying which genes are assigned to known proteins". This query must be executed in a database containing data about genes and proteins. The *Pfam* (see Table 2) is the database that provides the support for this query. Thus, by *R* programming, 45 genes are selected. Finally, each one of these 45 genes is verified with respect to its association to the Alzheimers disease. The biological entities which are relevant to this query are *Gene* and *Disease* (Task 5). Take into account only relevancy quality dimension, four databases provides proper support: *OMIM*, *Phenopedia*, *BioMart* and *ArrayExpress Archive*. The *OMIM* does not have an access

Table 2. Examples of molecular biology databases

Database	URL	Content
Array Express Archive	http://www.ebi.ac.uk/arrayexpress	Gene Disease Assay Microarray
Pfam database	http://pfam.sanger.ac.uk	Gene Protein Protein Function Protein Conformation
Gene Ontology	http://www.geneontology.org/	Gene Gene Function
BioMart	http://www.biomart.org	Gene Gene Function Protein Protein Function Disease
Phenopedia	http://www.hugenavigator.net	Gene Disease
OMIM	http://www.ncbi.nlm.nih.gov/omim	Gene Disease

mechanism, where the user informs the name of a disease and receives genes related to it. The same situation is observed for the *BioMart*. The *ArrayExpress* database returns raw Assays. The *Phenopedia* provides several meta-analysis studies in which genes are ranked properly according to a disease phenotype, i.e. Alzheimers disease. Therefore, the *Phenopedia* is the best database for this particular task. This last task demonstrates that a richer description of access mechanism of databases can help users in the database selection process.

6 Final Remarks

In the present work, the *BioDBCore* metadata has been analyzed and a prototype of an ontology has been defined, where some quality rules are defined for different quality dimensions using database metadata. The aim is to facilitate the searching process in a database catalog. In the future, we intent to improve this initial ontology. In this sense, the *Molecular Biology Summary Data* ontology, for example, has been defined to illustrate the approach, but it is necessary to improve this ontology and to verify, by experiments, the usefulness of it. Thinking in a scenario where *BioDBCore* data will be available in a near future (nowadays some of these data are available[1]), it will be possible to implement a real application using the proposed ontology.

[1] `http://www.biosharing.org/biodbcore`

Acknowledgments. This work is partially supported by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Brazil. The work of V. Burriel and O. Pastor has been developed with the support of MICINN and GVA under the projects PROS-Req TIN2010-19130-C02-02 and ORCA PROMETEO/2009/015, and co-financed with ERDF and Cátedra Tecnologías para la Salud of Universitat Politècnica de València financed by INDRA Systems.

References

1. Galperin, M.Y., Rigden, D.J., Fernndez-Surez, X.M.: The, nucleic acids research database issue and an updated nar online molecular biology database collection. Nucleic Acids Research 42(D1), D1–D6 (2014)
2. Gaudet, P., Bairoch, A., Field, D., Sansone, S., Taylor, C., Attwood, T., Bateman, A., Blake, J., Bult, C., Cherry, J., et al.: Towards BioDBcore: a community-defined information specification for biological databases. Nucleic Acids Research 39(suppl 1), D7 (2011)
3. Smedley, D., Schofield, P., Chen, C., Aidinis, V., Ainali, C., Bard, J., Balling, R., Birney, E., Blake, A., Bongcam-Rudloff, E.: et al.: Finding and sharing: new approaches to registries of databases and services for the biomedical sciences. Database: The Journal of Biological Databases and Curation 2010 (2010)
4. Naumann, F., Rolker, C.: Do metadata models meet iq requirements. In: Proceedings of the International Conference on Information Quality (IQ), pp. 99–114 (1999)
5. Wang, R.Y., Strong, D.M.: Beyond accuracy: what data quality means to data consumers. J. Manage. Inf. Syst. 12(4), 5–33 (1996)
6. Naumann, F., Leser, U., Freytag, J.C.: Quality-driven integration of heterogenous information systems. In: VLDB 1999: Proceedings of the 25th International Conference on Very Large Data Bases, San Francisco, CA, USA, pp. 447–458. Morgan Kaufmann Publishers Inc. (1999)
7. Lichtnow, D., Levin, A., Alves, R., Castello, I.M., Dopazo, J., Pastor, O., de Oliveira, J.P.M.: Using metadata and web metrics to create a ranking of genomic databases. In: White, B., Isaas, P., Santoro, F.M. (eds.) WWW/Internet 2011 Conference, IADIS,, pp. 253–260. IADIS Press (2011)
8. Lichtnow, D., Alves, R., de Oliveira, J.P.M., Levin, A., Pastor, O., Castello, I.M., Dopazo, J.: Using papers citations for selecting the best genomic databases. In: Proceedings of the 2011 30th International Conference of the Chilean Computer Science Society, SCCC 2011, pp. 33–42. IEEE Computer Society, Washington, DC (2011)
9. Cohen-Boulakia, S., Lair, S., Stransky, N., Graziani, S., Radvanyi, F., Barillot, E., Froidevaux, C.: Selecting biomedical data sources according to user preferences. Bioinformatics 20(1), 86–93 (2004)
10. O'Connor, M.F., Knublauch, H., Tu, S., Grosof, B.N., Dean, M., Grosso, W., Musen, M.A.: Supporting rule system interoperability on the semantic web with SWRL. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 974–986. Springer, Heidelberg (2005)
11. Martello, C.L., Alves, R.: Explorando marcadores gênicos em estudos transcriptomicos relacionados ao alzheimer. Technical report, Instituto de Biociências - UFRGS (2011)

Structural Comparative Analysis of Secreted NTPDase Models of *Schistosoma mansoni* and *Homo sapiens*

Vinicius Carius de Souza[1,⋆], Vinicius Schmitz Nunes[2,⋆], Eveline Gomes Vasconcelos[3], Priscila Faria-Pinto[3], and Priscila V.S.Z. Capriles[2]

[1] Departamento de Biologia, Instituto de Ciências Biológicas
[2] Faculdade de Engenharia, Instituto de Ciências Exatas
[3] Departamento de Bioquímica, Instituto de Ciências Biológicas, Universidade Federal de Juiz de Fora, MG, Brazil
{carius.souza,vschmitzn,priscilacapriles}@gmail.com, {eveline.vasconcelos,priscila.faria}@ufjf.edu.br

Abstract. The control of extracellular nucleoside concentrations by Nucleoside Triphosphate Diphosphohydrolase (NTPDase) is essential in the regulation of the purinergic signalling and also in immune response. In humans, eight members (HsNTPDase) were identified as transmembrane and secreted proteins. In *Schistosoma mansoni*, the causative agent of schistosomiasis, NTPDases similar to the humans enzymes have also been identified. The expression of these enzymes in *S. mansoni* (SmATPDases) is related to the weakening of the immune and inflammatory responses of the host against infections. Despite of the high phylogenetic conservation between these proteins, SmATPDases have been reported as molecular target candidates for antischistosomal treatment. In this work, we constructed three-dimensional models for secreted SmATPDase and HsNTPDase6, using comparative modeling technique. The comparative structural analysis aim the investigation of possible differences that could help future works in the development of new therapies that minimize the risk of cross inhibition.

Keywords: Nucleoside Triphosphate Diphosphohydrolases (NTPDase), *Schistosoma mansoni*, Three-dimensional models, Structural comparison.

1 Introduction

Nucleoside Triphosphate Diphosphohydrolases (NTPDases) are a family of enzymes that hydrolyze nucleoside di- and triphosphates to their corresponding mononucleotides, having divalent cations as cofactors, especially Ca^{2+} and Mg^{2+}. Structurally, these proteins have five (in some subgroups six) highly conserved regions named ACR (Apyrase Conserved Regions), which contribute to

⋆ These authors contributed equally to this work and should be considered co-first authors.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 91–98, 2014.

substrate binding and thus promote the hydrolysis of nucleotides[5]. The hydrolytic activity of NTPDase participates in the control of the nucleotides and nucleosides concentrations present in the extracellular environment, acting directly in the regulation of purinergic P2X and P2Y receptors, which are related to the dynamic of ion channels (Ligand-gated ion channel) and G-protein signaling[10], among other functions.

In humans and other mammals, eight members of NTPDases family have been identified: (i) six membrane proteins (NTPDases 1-4, 7 and 8) connected by two transmembrane helices (TM) and (ii) two secreted proteins (NTPDases 5 and 6) presenting a single TM domain that is cleaved. NTPDases 1-3 and 8 are present on the cell surface with size of approximately 500 amino acid residues. The extracellular domain (ECD) has ten conserved cysteines and, in addition to the five ACR present in NTPDases, has four more conserved regions. NTPDases 4 and 7, unlike the previous ones, are intracellular proteins presenting only four conserved cysteine in the ECD. NTPDases 5 and 6 are presented in soluble and membrane-bound forms (in endoplasmic reticulum and Golgi, respectively)[5]. NTPDase5 is expressed in macrophages, liver, kidney, prostate, colon and testis and NTPDase6 is highly expressed in heart[16].

The NTPDases have been identified of human parasites, such as *Toxoplasma gondii*, *Trypanosoma spp.*, *Leishmania sp.*, and *Schistosoma mansoni*. The expression of these enzymes in parasites is related to the weakening of the immune and inflammatory responses of the host against infections, and thus being a potential therapeutic target[11,12].

In *Schistosoma mansoni*, the main causative agent of Schistosomiasis in South America, two isoforms of NTPDases were identified: (i) NTPDase type 1 (SmATPDase1) which is expressed in the adult worm tegument and has two TM, and (ii) NTPDase type 2 (SmATPDase2) that is expressed and secreted in both adults and eggs. The presence of these enzymes seems to be related with the control of the concentration of purine molecules responsible for signaling the clotting around the parasite, since he lives within blood vessels[1,7].

Studies have shown a phylogenetic relationship between SmATPDases and mammalian NTPDases. More precisely, the SmATPDase1 is related to NTPDases 1-3 and 8 of mammals, while the SmATPDase2 is more related to NTPDases 5 and 6[2,10]. The literature suggests that SmATPDases were involved in the evasion of the host defence, due to the location of these isoforms in the parasite and the importance of nucleoside di- and triphosphates in activating cells of the host immune system[1,7,12]. It is important to note that some experiments already demonstrated the inhibition of the enzymatic activity of SmATPDases by alkylaminoalkanethiosulfuric acids[9]. Such assumptions reinforce the importance of SmATPDases as molecular target candidates for antischistosomal treatment[3].

The objective of this work is to build the three-dimensional (3D) models of the secreted NTPDase of *S. mansoni* (SmATPDase2) and human (HsNTPDase6).

The comparative analysis between these isoforms is critical to understand their differences aiming to the development of new therapies that minimize the risks of cross inhibition.

2 Methodology

2.1 Detection of Transmembrane Domain and Signal Peptide

The prediction of TM domains was performed using the programs Phobius, TMHMM, HMMTOP and TMpred. The investigation of putative regions of signal peptide and cleavage sites was performed using the predictor SignalPv4.0. The molecular weight of the predicted secretable domain (ECD) of both proteins was calculated using the program ProtParam, in order to corroborate our results with those previously described in the literature[4,8].

2.2 Three-Dimensional Protein Model Construction

Templates for 3D model construction were selected from Protein Data Bank (PDB) based on local alignment (via BlastP) between deposited proteins and target sequences, 3D structures were selected based on amino acid sequence identity, similarity, coverage, enzymatic activity and on the presence of ligands.

Multiple sequence alignment between SmATPDase2 and HsNTPDase6 with templates was generated using the program ClustalΩ. To verify the consensus areas between target sequences and the secondary structure of templates, the following predictors of secondary structure were used: PSIPRED, APSSP2, Jpred3, PSSPRED, Sable and Jufo9D. Due to absence of template in certain regions of both target proteins, secondary structure restraints (according to the results of the predictors) were used during the 3D comparative modeling construction using the program Modeller9v13.

All constructed models were assessed using the programs Procheck and Molprobity for the analysis of stereochemical quality. The energy values were analyzed using the nDOPE and Molpdf methods calculated by Modeller9v13.

2.3 SAS, Volume and Electrostatic Profile

Additionally to the comparative structural analyses, overall and catalytic site differences between the electrostatic profile, volume and solvent accessible surface area (SASA) of SmATPDase2 and HsNTPDase6 were evaluated. The analyses of the overall volume and SASA were performed using the plugin VolArea in the VMD program and for the active sites was used the program CASTp. The calculation and analyses of the electrostatic potentials were performed using the programs PDB2PQR v1.9 and APBS v1.4.1.

3 Results and Discussion

3D structures of both SmATPDase2 and HsNTPDase6 were constructed by comparative modeling using as templates the crystal structures of Rattus norvegicus: RnNTPDase1 (PDB 3ZX3 - chain A)[14] and RnNTPDase2 (PDB 4BR0)[13]. Sequence identity (and similarity) between SmATPDase2 (UniProt A1BXT9) and 3ZX3 and 4BR0 were 33%(47%) and 26%(41%), respectively. Whereas for the HsNTPDase6 (UniProt ID: O75354) these values were 36%(51%) and 32%(47%). Primary and secondary structural alignments are presented in Figure 1.

The characterization of RnNTPDases 1 and 2 shows that, although only the ECD domains have been crystallized, both are transmembrane enzymes[13,15] with only one TM. The results of TM predictors have also confirmed the presence of a single TM in the SmATPDase2 and HsNTPDase6 (Table 1).

Table 1. Prediction of transmembrane regions in SmATPDase2 and HsNTPDase6

Enzyme	Transmembrane Prediction			
	Phobius	TMHMM	TMpred	HMMTOP
SmATPDase2	63–81	63–80	61–82	63–80
HsNTPDase6	38–59	38–60	40–60	43–60

The prediction of cleavage sites using SignalP v4.0 presented the segment M1–N83 as a probable cleavage region in SmATPDase2. In HsNTPDase6 we define as the cleavage region the segment M1–R76, according to the literature data[4]. The analysis with the program ProtParam resulted in the following molecular weights after cleavage: ∼55kDa for SmATPDase2 and ∼45kDa for HsNTPDase6. These results are consistent with previous studies[4,8].

The SmATPDase2 and HsNTPDase6 models were generated considering the presence of ligands in the active site of 4BR0 structure: AMPNP (Adenosine 5-(alpha, beta-imido) diphosphate), analogous to ADP, and calcium ion. According to the literature and the analysis of the structural alignment of the templates, we considered the presence of six conserved water molecules: (i) four responsible for stabilizing the metal ion, (ii) one responsible for nucleophilic attack, and (iii) one positioned next to the last phosphate[15]. Water molecules (O2003, O2005, O2046, O2069, O2070 and O2101) were added from 4BR0 structure. Literature has shown that such waters and divalent ions are required for the process of hydrolysis of nucleotide[13,15]. Structural alignment between targets and templates is presented in the Figure 2.

Catalytic residues present in ACR are well conserved between SmATPDase2 and HsNTPDase6 models and their templates. According to Zebisch & Sträter (2008), there are twenty main interacting residues in the catalytic site of RnNTPDases2, of which eight residues (D45, A123, E165, T122, D201, S206, Q208 and W436) interact with six conserved waters described, and four residues (R245, A347, Y350 and R394) forming hydrogen bonds with the ligand enabling its docking to the catalytic site. The mapping of these residues in both models, from the structural superposition with 4BR0, is listed in Table 2. Residues that form hydrogen bonds with ligand were also mapped (Table 2).

Fig. 1. Primary and secondary structural alignment between SmATPDase2, HsNTPDase6 and selected templates from *R. novergicus* (RnNTPDase1 - 3ZX3 and RnNTPDase2 - 4BR0). Strands are colored in yellow and helices are represented by red rectangles. The five Apyrase Conserved Regions (ACR) are marked with black frames. The sequence alignment was performed using ClustalΩ and secondary structure description was obtained via program Stride.

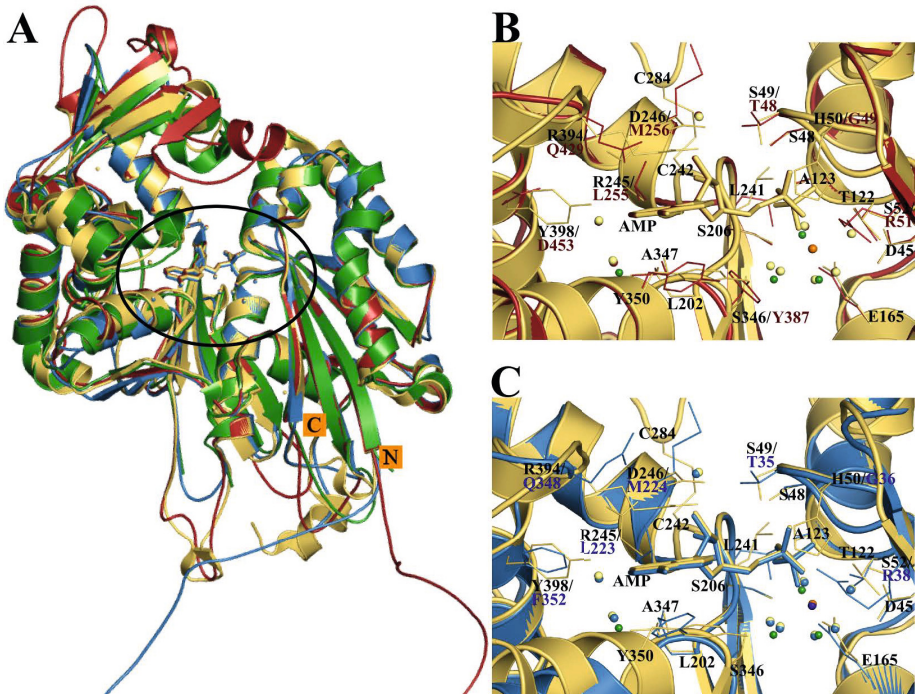


Fig. 2. 3D structural comparison between SmATPDase2 and HsNTPDase6 models and selected templates. (A) Structural superposition between SmATPDase2 (red) and HsNTPDase6 (blue) models and the templates RnNTPDase1 (3ZX3 chain A - green) and RnNTPDase2 (4BR0 - yellow). 3D structures are presented in Cartoon, ligand (AMPNP), Ca^{2+} and water molecules are presented in Licorice, and the active site are marked with a black circle. In detail are presented the active site comparison between RnNTPDase2 and (B) SmATPDase2 and (C) HsNTPDase6. Residue numbering according to RnNTPDase2 (black), SmATPDase2 (red) and HsNTPDase6 (blue). Ca^{2+} from SmATPDase2 is presented in dark blue, from HsNTPDase6 in pink and from RnNTPDase2 in orange.

Table 2. Active site comparison of SmATPDase2 and HsNTPDase6 models

Interations with	SmATPDase2	HsNTPDase6
Waters	D44(ACR1), A123(ACR2), E164(ACR3), D203(ACR4), S208(ACR4), Q210(ACR4), W469(ACR5).	D31(ACR1), A107(ACR2), E148(ACR3), D175(ACR4), S180(ACR4), Q182(ACR4), W388(ACR5).
AMPNP	L255, A388, Y391, Q429	L233, Y308, D311, Q348

Analysis of the overall structures and active sites of the SmATPDase2 and HsNTPDase6 models showed that, despite having similar values of the electrostatic potential, the SmATPDase2 is significantly higher than HsNTPDase6 (in relation to the volume and SASA). The results are shown in Table 3.

Table 3. Structural comparison of SmATPDase2 and HsNTPDase6 models

Enzyme	Total			Active Site		
	Volume[a]	SASA[b]	Electrostatic[c]	Volume[a]	SASA[b]	Electrostatic[c]
SmATPDase2	83,569	25,149.36	3.78E+05	889	581.34	2.27E+04
HsNTPDase6	69,163	21,617.55	3.08E+05	1,137	610.43	2.88E+04

[a]Volume in $\AA^3$. [b]SASA in $\AA^2$. [c]Electrostatic in kJ/mol.

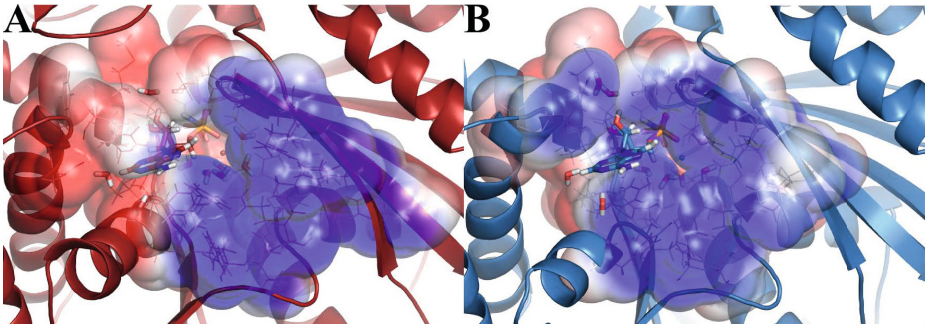


Fig. 3. Comparative analysis between electrostatic profile of SmATPDase2 and HsNTPDase6 active sites. (A) SmATPDase2 (red). (B) HsNTPDase6 (blue). 3D structures are presented in Cartoon, ligand (AMPNP), Ca^{2+} and water molecules are presented in Licorice. The electrostatic potential vary from -1 (red) to 1 (blue) kT/e.

4 Conclusions

In this paper, we presented the secreted NTPDase 3D models of *Schistosoma mansoni* (SmATPDase2) and human (HsNTPDase6), constructed by comparative modeling. Since they do not have structural characterization and, despite being phylogenetically related, have selective inhibition for certain compounds[9], the generation of these 3D models becomes very important once it may help future works of structure-based drug design.

We observed that the two generated models have high structural similarity, mainly in regions considered important for the hydrolysis of the ligand. However, the active site of SmATPDase2 has larger volume and SASA than the HsNTPDase6 and, despite they present similar electrostatic potencial, the electrostatic profile of SmATPDase2 is more negatively charged in the region of nucleoside stabilization. Unlike the RnNTPDases, both SmATPDase2 and HsNTPDase6 do not present in this region an important disulfide bond (C242-C284 from 4BR0) involved in the enzyme stabilization.

Considering structural similarities, in future works we intend to investigate other cavities of these proteins that could indicate alternative sites for the inhibition of SmATPDase2. These regions will be analyzed using molecular dynamics, normal modes and protein-ligand docking.

Acknowledgments. This work was supported by CAPES.

References

1. DeMarco, R., et al.: Molecular characterization and immunolocalization of *Schistosoma mansoni* ATP-diphosphohydrolase. Biochemical and Biophysical Research Communications 307(4), 831–838 (2003)
2. Faria-Pinto, P., et al.: Mapping of the conserved antigenic domains shared between potato apyrase and parasite ATP diphosphohydrolases: potential application in human parasitic diseases. Parasitology 135(8), 943–953 (2008)
3. Fonseca, C.T., et al.: Schistosoma tegument proteins in vaccine and diagnosis development: an update. Journal of Parasitology Research, 541268 (2012)
4. Ivanenkov, V.V., et al.: Bacterial expression, characterization, and disulfide bond determination of soluble human NTPDase6 (CD39L2) nucleotidase: implications for structure and function. Biochemistry 42(40), 11726–11735 (2003)
5. Knowles, A.F.: The GDA1_CD39 superfamily: NTPDases with diverse functions. Purinergic Signalling 7(1), 21–45 (2011)
6. Kukulski, F., Lvesque, S.A., Svigny, J.: Impact of ectoenzymes on p2 and p1 receptor signaling. Advances in Pharmacology (61), 263–299 (2011)
7. Levano-Garcia, J., et al.: Characterization of *Schistosoma mansoni* ATPDase2 gene, a novel apyrase family member. Biochemical and Biophysical Research Communications 352(2), 384–389 (2007)
8. Mendes, R.G.P.R., et al.: Immunostimulatory property of a synthetic peptide belonging to the soluble ATP diphosphohydro lase isoform (SmATPDase 2) and immunolocalisation of this protein in the *Schistosoma mansoni* egg. Memórias do Instituto Oswaldo Cruz 106(7), 808–813 (2011)
9. Penido, M.L.O., et al.: A new series of schistosomicide drugs, the alkylaminoalkanethiosulfuric acids, partially inhibit the activity of *Schistosoma mansoni* ATP diphosphohydrolase. European Journal of Pharmacology 570(1-3), 10–17 (2007)
10. Sansom, F.M.: The role of the NTPDase enzyme family in parasites: what do we know, and where to from here? Parasitology 139(8), 963–980 (2012)
11. Sansom, F.M., Robson, S.C., Hartland, E.L.: Possible effects of microbial ectonucleoside triphosphate diphosphohydrolases on host-pathogen interactions. Microbiology and Molecular Biology Reviews 72(4), 765–781 (2008)
12. Vasconcelos, E.G., et al.: Partial purification and immunohistochemical localization of ATP diphosphohydrolase from *Schistosoma mansoni*. Immunological cross-reactivities with potato apyrase and *Toxoplasma gondii* nucleoside triphosphate hydrolas. Journal of Biological Chemistry 271(36), 22139–22145 (1996)
13. Zebisch, M., et al.: Crystallographic Snapshots along the Reaction Pathway of Nucleoside Triphosphate Diphosphohydrolases. Structure 21(8), 1460–1475 (2013)
14. Zebisch, M., et al.: Crystallographic evidence for a domain motion in rat nucleoside triphosphate diphosphohydrolase (NTPDase) 1. Journal of Molecular Biology 415(2), 288–306 (2012)
15. Zebisch, M., Strater, N.: Structural insight into signal conversion and inactivation by NTPDase2 in purinergic signaling. PNAS 105(19), 6882–6887 (2008)
16. Zimmermann, H., Zebisch, M.: Cellular function and molecular structure of ectonucleotidases. Purinergic Signalling 8(3), 437–502 (2012)

Length and Symmetry on the Sorting by Weighted Inversions Problem

Christian Baudet[1], Ulisses Dias[2], and Zanoni Dias[2]

[1] Université Lyon I, INRIA Bamboo Team, France
christian.baudet@univ-lyon1.fr
[2] University of Campinas, Institute of Computing, Brazil
{udias,zanoni}@ic.unicamp.br

Abstract. Large-scale mutational events that occur when stretches of DNA sequence move throughout genomes are called genome rearrangement events. In bacteria, inversions are one of the most frequently observed rearrangements. In some bacterial families, inversions are biased in favor of symmetry as shown by recent research [6, 8, 10]. In addition, several results suggest that short segment inversions are more frequent in the evolution of microbial genomes [4,6,15]. Despite the fact that symmetry and length of the reversed segments seem very important, they have not been considered together in any problem in the genome rearrangement field. Here, we define the problem of sorting genomes (or permutations) using inversions whose costs are assigned based on their lengths and asymmetries. We present five procedures and we assess these procedure performances on small sized permutations. The ideas presented in this paper provide insights to solve the problem and set the stage for a proper theoretical analysis.

1 Introduction

Among various large-scale rearrangement events that have been proposed to date, inversions were established as the main explanation for the genomic divergence in many organisms [6, 8, 11]. An inversion occurs when a chromosome breaks at two locations, and the DNA between those locations is reversed.

In some families of bacteria, an 'X'-pattern is observed when two circular chromosomes are aligned [8,10]. Inversions symmetric to the origin of replication (meaning that the breakpoints are equally distant from the origin of replication) have been proposed as the primary mechanism that explains the pattern [10]. The justification relies on the fact that one single highly asymmetric inversion affecting a large area of the genome could destroy the 'X'-pattern, although short inversions may still preserve it.

Darlink, Miklós and Ragan [6] studied eight *Yersinia* genomes and added evidence that symmetric inversions are "over-represented" with respect to other types of inversions. They also found that inversions are shorter than expected under a neutral model. In many cases, short inversions affect only a single gene, as observed by Lefebvre *et al.* [12] and Sankoff *et al.* [16], which contrasts with

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 99–106, 2014.

the null hypothesis that the two endpoints of an inversion occur by random and independently.

Despite the importance of symmetry and length of the reversed segment, both have been somewhat overlooked in the genome rearrangement field. Indeed, the most important result regarding inversions is a polynomial time algoritm presented by Hannenhalli and Pevzner [11] that considers an unit cost for each inversion no matter its length or symmetry. When gene orientation is not taken into account, finding the minimum number of inversions that transform one genome into the other is a NP-Hard problem [5].

Some results have considered at least one of the concepts. There is a research line that considers the total sum of the inversion lengths as the objective function of a minimization problem. Several results have been presented both when gene orientation is considered [2, 17] and when it is not [1, 3, 14]. Recently, Arruda *et al.* [2] developed a randomized approach that starts with a scenario with the minimum number of inversions, but allows the number of inversions to increase if the outcome is a reduction in the total sum of the inversion lengths.

Regarding symmetry, the first results were presented by Ohlebusch *et al* [13]. Their algorithm uses symmetric inversions in a restricted setting to compute an ancestral genome and, therefore, is not a generic algorithm to compute the rearrangement distance using only symmetric inversions. In 2012, Dias *et al.* presented an algorithm that considers only symmetric and almost-simmetric inversions [9]. They later included unitary inversions to the problem and provided a randomized heuristic to compute scenarios between two genomes that uses solely these operations [7].

Here we propose a new genome rearrangement problem that combines the concepts of symmetry and length of the reversed segments. Whereas previous works restricted the set of allowed operations by considering only inversions that satisfy constrains like symmetry or almost-symmetry [7, 9], here we allow all possible inversions. The problem we are proposing aims at finding low-cost scenarios between genomes when gene orientation is not taken into account. The results obtained are the first steps in exploring this interesting new problem.

2 Definitions

Formally, a chromosome is represented as a n-tuple whose elements represent genes. If we assume no gene duplication, then this n-tuple is a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$, $1 \leq \pi_i \leq n$ and $\pi_i \neq \pi_j \leftrightarrow i \neq j$. Because we focus on bacterial chromosomes, we assume permutations to be circular, and π_1 is the first gene after the origin of replication.

The inverse of a permutation π is denoted by π^{-1}, for which $\pi^{-1}_{\pi_i} = i$ for all $1 \leq i \leq n$. The composition between two permutations π and σ is similar to function composition in such way that $\pi \cdot \sigma = (\pi_{\sigma_1}\ \pi_{\sigma_2}\ \ldots\ \pi_{\sigma_n})$.

An inversion $\rho(i, j)$, $1 \leq i \leq j \leq n$, is a rearrangement that transforms π into $\pi \cdot \rho(i, j) = (\pi_1\ \ldots\ \pi_{i-1}\ \underline{\pi_j\ \pi_{j-1}\ \ldots\ \pi_{i+1}\ \pi_i}\ \pi_{j+1}\ \ldots\ \pi_n)$.

Given two permutations α and σ, the inversion distance $d(\alpha, \sigma)$ is the size t of the minimum sequence of operations $\rho_1, \rho_2, \ldots, \rho_t$ such that $\alpha \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \rho_t = \sigma$.

Let $\iota = (1\ 2\ \ldots\ n)$ be the identity permutation, sorting a permutation $\pi = (\pi_1\ \pi_2\ \ldots\ \pi_n)$ is the process of transforming π into ι and the inversion distance between them is denoted as $d(\pi, \iota) = d(\pi)$. Note that $\sigma \cdot \sigma^{-1} = \sigma^{-1} \cdot \sigma = \iota_n$. Therefore, the inversion distance $d(\alpha, \sigma)$ is equivalent to transform a permutation π into a permutation ι if we take $\pi = \sigma^{-1} \cdot \alpha$, because $d(\alpha, \sigma) = d(\sigma^{-1} \cdot \alpha, \sigma^{-1} \cdot \sigma) = d(\pi, \iota) = d(\pi)$. Therefore, we hereafter consider the sorting by inversions problem which aims at finding the sorting distance $d(\pi)$ for an arbitrary permutation π.

The following functions can be applied to identify any element i in the permutation π. **Position**: $p(\pi, i) = k \Leftrightarrow |\pi[k]| = i$, $p(\pi, i) \in \{1, 2, \ldots, n\}$. **Slice**: $slice(\pi, i) = \min\{p(\pi, i), n - p(\pi, i) + 1\}$, $slice(\pi, i) \in \{1, 2, \ldots, \lceil \frac{n}{2} \rceil\}$.

We will now define the cost function for each inversion $\rho(i, j)$ and then we explain two cases that arise. Our cost function is given by $cost(\rho(i, j)) = |slice(\iota, i) - slice(\iota, j)| + 1$. Figure 1 illustrates both cases.

Case 1: $i, j \leq \lceil \frac{n}{2} \rceil$ or $i, j \geq \lceil \frac{n}{2} \rceil$.
In this case, the cost function can be simplified to $cost(\rho(i, j)) = abs(i-j)+1$, which means that it is proportional to the number of elements in the reversed segment. This cost is what one would expect from a length-weighted inversion distance in such a way that larger inversions cost more than short inversions.

Case 2: $i > \lceil \frac{n}{2} \rceil$ and $j < \lceil \frac{n}{2} \rceil$, or $j > \lceil \frac{n}{2} \rceil$ and $i < \lceil \frac{n}{2} \rceil$.
In this case, the cost function is penalizing the asymmetry instead of the number of elements in the reversed segment. In effect, if the inversion $\rho(i, j)$ is perfectly symmetric (meaning that i and j are equally distant from the origin of replication), then the cost is given by $cost(\rho(i, j)) = 1$.

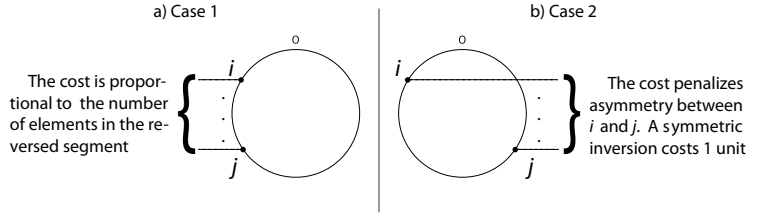


Fig. 1. Effect of the cost function when (a) $i, j \leq \lceil \frac{n}{2} \rceil$ or $i, j \geq \lceil \frac{n}{2} \rceil$ and (b) $i > \lceil \frac{n}{2} \rceil$ and $j < \lceil \frac{n}{2} \rceil$, or $j > \lceil \frac{n}{2} \rceil$ and $i < \lceil \frac{n}{2} \rceil$

3 Algorithms

This section presents several algorithms that take advantage from the characteristics of the cost functions.

3.1 Greedy Function

This algorithm uses a greedy function in order to estimate how good an inversion might be. The greedy function uses two concepts: breakpoints and slice-misplaced pairs.

Definition 1. *Breakpoints: let us extend the permutation π to include the elements $\pi_0 = 0$ and $\pi_{n+1} = n + 1$. The pair π_i and π_{i+1}, for $0 \leq i \leq n$, is a breakpoint if $|\pi_{i+1} - \pi_i| \neq 1$. We use $b(\pi)$ to represent the number of breakpoints in a permutation and $\Delta_b(\pi, \rho) = b(\pi \cdot \rho) - b(\pi)$ to represent the variation in the number of breakpoints caused by an inversion ρ.*

Definition 2. *Slice-Misplaced pairs: let π_i, π_j be two elements in π such that $slice(\pi, \pi_i) > slice(\pi, \pi_j)$. We say that π_i, π_j corresponds to a slice-misplaced pair if $slice(\iota, \pi_i) < slice(\iota, \pi_j)$. We use $m(\pi)$ to represent the number of slice-misplaced pairs in π and $\Delta_m(\pi, \rho) = m(\pi \cdot \rho) - m(\pi)$ to represent the variation in the number of misplaced-pairs caused by an inversion ρ.*

The identity permutation ι is the only permutation with no breakpoints. Therefore, an inversion that decreases the number of breakpoints indirectly leads up to the identity permutation. The identity permutation also has no misplaced pairs, so we will create a function that combines both concepts in order to estimate how good an inversion is.

Since an inversion can affect only two breakpoints, we know that $\Delta_b(\pi, \rho) \in \{-2, -1, 0, 1, 2\}$. The variation in the number of slice-misplaced pairs Δ_m is not well-behaved like Δ_b. Thus, we decided to favour breakpoints reduction in our greedy function: $h(\pi, \rho) = \Delta_b(\pi, \rho) + \frac{\Delta_m(\pi,\rho)}{n^2}$. Therefore, the benefit of an inversion is given by $\delta(\pi, \rho) = \frac{h(\pi,\rho)}{cost(\rho)}$. We construct a sequence of inversions that sorts π by iteratively adding an inversion with the best benefit among all possible inversions.

3.2 Left or Right Heuristic

We first divide the elements in π in two groups. The first group refers to the elements that are in slices classified as `sorted` and the second group comprises those elements that are in `unsorted` slices. A slice s is in the `sorted` group if $p(\pi, s) = s$, $p(\pi, n - s + 1) = n - s + 1$, and the slices $\{1, 2, \ldots, s - 1\}$ are also in the `sorted` group. Otherwise, s is in the `unsorted` group.

First, the *Left or Right* heuristic selects the least slice in the `unsorted` group. Then, we determine the element that should be moved first to that slice: the left or the right. The left (right) side is composed of the elements which are in positions that have indices bigger (lower) than the middle position.

To make this choice, we use an auxiliary function f_1 which determines the total weight to put a given element in its right place. The function considers only inversions ρ such that $cost(\rho) \leq 2$. The minimum number of such inversions that is necessary to move the element onto its right position is counted and the weight is computed.

If the slice has only one element that does not belong to it, we just find the element that should be in that slice and perform an inversion to place it closer to its right position. We use inversions ρ whose $cost(\rho) \leq 2$ following the principle of always applying non-expensive movement.

If the slice has two elements that are misplaced, we compute the total weight to place the left element and the total weight to place the right element. Then, we choose the element that has the smallest total weight. In case of tie, we move the right element.

3.3 Lock Heuristic

This heuristic adds a new step that will be called before using the *Left or Right* heuristic. We first search for a `lock` inversion, which puts an element directly into its final position in the `unsorted` slice with the least value.

We also consider the possibility of `indirect lock` inversions that put the element in its final position in two steps: first, it moves the element to its final slice, but in the opposite side. Then, it moves the element to its right place with a perfectly symmetric inversion. In the case where the opposite side is already `locked` (opposite side has already the right element), we consider moving the element one slice above and then applying an almost-symmetric inversion.

An auxiliary function f_2 is used to evaluate the inversions. This function takes the elements π_i and π_j that are in the endpoints of the inversion $\rho(i, j)$ and uses the function f_1 to compute the total weight to put them in their final positions. We compute the total weight before and after applying the inversions and compute the gain. If the gain is positive (*i.e.*, the total weight decreased) and this gain is equal to or bigger than the inversion weight, $\rho(i, j)$ is considered a valid inversion.

The heuristic evaluates the inversions for placing the right and left elements, and then applies the inversion with least cost. If a lock inversion does not exist, then the *Left or Right* heuristic is used.

3.4 Reduce Heuristic

This heuristic computes all possible inversions that remove at least one breakpoint and is considered valid by the function f_2 as described in Section 3.3. We filter the inversions having minimum weight according to f_2, and if more than one inversion remains, we select one that removes two breakpoints. If no inversion reduces the number of breakpoints, we apply the *Left or Right* heuristic (Section 3.2). This new heuristic is called *Reduce* heuristic.

It is also possible to combine the *Reduce* heuristic with the *Lock* heuristic to generate other approach called *Lock or Reduce*. This new approach searches for a valid lock inversion ρ_1 and for a valid inversion ρ_2 which reduces the number of breakpoints. If just one of ρ_1 or ρ_2 exists, the heuristic simply applies it. If both exist, the heuristic selects the one that has the smallest weight (in case of tie, apply ρ_1). Finally, if none exists, the *Left or Right* heuristic is used.

4 Experimental Results

We generated a dataset with all possible instance π, such that $|\pi| \leq 12$. Since the permutations in the dataset are small, we were able to compute a minimum cost solution for comparison purposes. The minimum cost solution was calculated using a graph structure G_n, for $n \in \{1, 2, \ldots, 12\}$. We define G_n as follows. A permutation π is a vertex in G_n iff π has n elements. Let π and σ be two vertices in G_n, we build an edge from (π, σ) iff there is an inversion ρ that transforms π into σ. The weight assigned to this edge is $cost(\rho)$. Finally, we calculate the shortest path from ι to each vertex in G_n using a variant of Dijkstra's algorithm for the single-source shortest-paths problem. This variant gives us the minimum cost to sort permutations in G_n, as well as an optimum scenario of inversions.

Let $heuristic\_cost(\pi)$ be the cost for a sorting sequence of π and $cost(\pi)$ be the optimum cost, we can compute the approximation ratio as $\frac{heuristic\_cost(\pi)}{cost(\pi)}$. The first graph in Figure 2 shows how often each heuristic returns the optimum cost. The second and third graphs exhibit, respectively, the average and maximum ratios observed among permutations of same size.

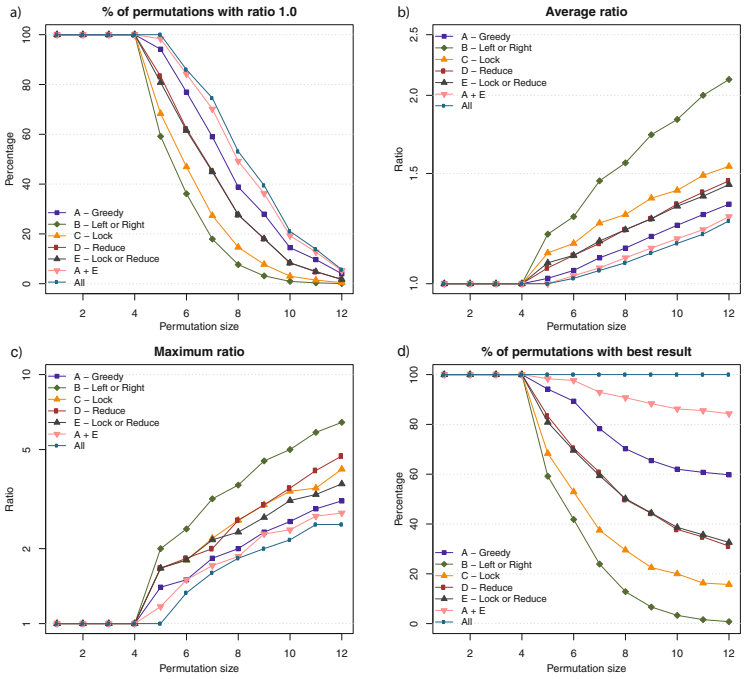


Fig. 2. In (a) we show how often each heuristic returns a minimum cost solution. In (b) and (c) we show the average and the maximum ratio, respectively. In (d) we show how often each heuristic succeeds in providing the best answer among all the heuristics.

The graphs in figures 2(a), 2(b) and 2(c) are maximum or average values. Thus, they may not answer the question: for a single instance π, is there any algorithm that is likely to provide the best answer? The fourth graph discuss this question by assessing the number of times each algorithm provides the least costly sequence. We observe that *Greedy* leads to the best results. In fact, this heuristic is consistently better than the other heuristics in every aspects we plot in Figure 2. The heuristics *Reduce* and *Lock or Reduce* are very similar when we consider how often each one returns the optimum cost and the average ratio. However, the *Lock or Reduce* heuristic seems more appropriate for those permutations that are hard to optimize, since the maximum ratio curve for *Lock or Reduce* (shown in the third graph) is significantly better than that for *Reduce*.

We added new curves to see if one could take advantage of using more than one heuristic other than *Greedy*. The curve labeled as `A+E` selects for each instance the less costly result between those produced by the *Greedy* heuristic and the *Lock or Reduce* heuristic. As we can see, using both heuristics is consistently better than using solely the *Greedy* heuristic in every aspect we are studying.

A final test checks if any profit is gained from running all possible heuristic, which is what we consider in the curve `All`. The increase in performance produced by `All` is consistent when compared to `A+E`. In every graph, the `All` curves are far from the others, which leads to the conclusion that using a combination of more than one heuristic accomplishes satisfactory results.

5 Conclusions

In this work, we have defined a new genome rearrangement problem based on the concepts of symmetry and length of the reversed segments in order to assign a non-unit cost for each inversion. The problem we are proposing aims at finding low-cost scenarios between genomes when gene orientation is not taken into account. We have provided the first steps in exploring this problem.

We presented five heuristics and we assessed their performances on small sized permutations. The ideas used in order to develop these heuristics together with the experimental results set the stage for a proper theoretical analysis.

As in other inversion sorting problems, we would like to know the complexity of determining the distance between any two genomes using only the operations we defined. That seems to be a difficult problem that we intend to keep studying. We plan to design approximation algorithms and more effective heuristics.

Acknowledgments. This work was supported by a Postdoctoral Fellowship from FAPESP to UD (number 2012/01584-3), by project fundings from CNPq to ZD (numbers 306730/ 2012-0, 477692/2012-5 and 483370/2013-4), and by French Project ANR MIRI BLAN08-1335497 and the ERC Advanced Grant SISYPHE to CB. The authors also thank the Center for Computational Engineering and Sciences at Unicamp for financial support through the FAPESP/CEPID Grant 2013/08293-7.

References

1. Arruda, T.S., Dias, U., Dias, Z.: Heuristics for the sorting by length-weighted inversion problem. In: Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics, pp. 498–507 (2013)
2. Arruda, T.S., Dias, U., Dias, Z.: Heuristics for the sorting by length-weighted inversions problem on signed permutations. In: Dediu, A.-H., Martín-Vide, C., Truthe, B. (eds.) AlCoB 2014. LNCS, vol. 8542, pp. 59–70. Springer, Heidelberg (2014)
3. Bender, M.A., Ge, D., He, S., Hu, H., Pinter, R.Y., Skiena, S., Swidan, F.: Improved bounds on sorting by length-weighted reversals. Journal of Computer and System Sciences 74(5), 744–774 (2008)
4. Blanchette, M., Kunisawa, T., Sankoff, D.: Parametric genome rearrangement. Gene 172(1), C11–C17 (1996)
5. Caprara, A.: Sorting permutations by reversals and Eulerian cycle decompositions. SIAM Journal on Discrete Mathematics 12(1), 91–110 (1999)
6. Darling, A.E., Miklós, I., Ragan, M.A.: Dynamics of genome rearrangement in bacterial populations. PLoS Genetics 4(7), e1000128 (2008)
7. Dias, U., Baudet, C., Dias, Z.: Greedy randomized search procedure to sort genomes using symmetric, almost-symmetric and unitary inversions. In: Proceedings of the International Conference on Bioinformatics, Computational Biology and Biomedical Informatics, BCB 2013, pp. 181–190. ACM, New York (2013)
8. Dias, U., Dias, Z., Setubal, J.C.: A simulation tool for the study of symmetric inversions in bacterial genomes. In: Tannier, E. (ed.) RECOMB-CG 2010. LNCS, vol. 6398, pp. 240–251. Springer, Heidelberg (2010)
9. Dias, Z., Dias, U., Setubal, J.C., Heath, L.S.: Sorting genomes using almost-symmetric inversions. In: Proceedings of the 27th Symposium On Applied Computing (SAC 2012), Riva del Garda, Italy, pp. 1–7 (2012)
10. Eisen, J.A., Heidelberg, J.F., White, O., Salzberg, S.L.: Evidence for symmetric chromosomal inversions around the replication origin in bacteria. Genome Biology 1(6), research0011.1–research0011.9 (2000)
11. Hannenhalli, S., Pevzner, P.A.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM 46(1), 1–27 (1999)
12. Lefebvre, J.F., El-Mabrouk, N., Tillier, E., Sankoff, D.: Detection and validation of single gene inversions. Bioinformatics 19(suppl 1), i190–196 (2003)
13. Ohlebusch, E., Abouelhoda, M.I., Hockel, K., Stallkamp, J.: The median problem for the reversal distance in circular bacterial genomes. In: Apostolico, A., Crochemore, M., Park, K. (eds.) CPM 2005. LNCS, vol. 3537, pp. 116–127. Springer, Heidelberg (2005)
14. Pinter, R.Y., Skiena, S.: Genomic sorting with length-weighted reversals. Genome Informatics 13, 2002 (2002)
15. Sankoff, D.: Short inversions and conserved gene cluster. Bioinformatics 18(10), 1305–1308 (2002)
16. Sankoff, D., Lefebvre, J.F., Tillier, E., Maler, A., El-Mabrouk, N.: The distribution of inversion lengths in bacteria. In: Lagergren, J. (ed.) RECOMB-WS 2004. LNCS (LNBI), vol. 3388, pp. 97–108. Springer, Heidelberg (2005)
17. Swidan, F., Bender, M., Ge, D., He, S., Hu, H., Pinter, R.: Sorting by length-weighted reversals: Dealing with signs and circularity. In: Sahinalp, S.C., Muthukrishnan, S.M., Dogrusoz, U. (eds.) CPM 2004. LNCS, vol. 3109, pp. 32–46. Springer, Heidelberg (2004)

Storage Policy for Genomic Data in Hybrid Federated Clouds

Ricardo Gallon, Maristela Holanda, Aletéia Araújo, and Maria E. Walter

Computer Science Department, Universidade de Brasília, Brasília, Brazil
rfgallon@gmail.com, {mholanda,aleteia,mia}@cic.unb.br

Abstract. Execution performance of bioinformatics workflows in cloud federated environments is strongly affected by data storage and retrieval, due to the large volumes of information in genomic sequences. This paper presents a storage policy for files used in a typical bioinformatics application with genomic data that aims to reduce their transfer time and then contribute to a faster execution of the workflow. We discuss a case study using the BioNimbuZ federated cloud platform. Our results show that this storage policy significantly improved times for transferring files, and thus lowered the total time to execute the workflow.

1 Introduction

Hundreds of genome projects around the world have to treat large amounts of genomic data produced by high-throughput sequencing machines. A single genome project generates gigabytes of data, which are analyzed by computational tools and may generate terabytes of storage. Genome projects, and bioinformatics analysis as a whole, are usually supported by workflows composed of tools and databases managed in physically separate institutions.

In recent years, the paradigm of cloud computing has been developed [1,2,4,3], so that users may transparently access a wide variety of distributed infrastructures and systems. In these clouds, both computing and storing data are treated so that it gives to the user the illusion that the amount of resources is unlimited. However, considering the continuous increasing of computational and storage power needed by different bioinformatics applications, developed in different environments, working with only one cloud may be restrictive for bioinformatics applications. Thus, federated cloud [5,6] is an interesting alternative to execute bioinformatics applications.

In this paper, we propose a storage policy for hybrid federated clouds that aims to choose the best resource to store or retrieve files used in bioinformatics workflows. We developed a case study in the BioNimbuZ [7,8] platform, a hybrid federated cloud computing platform that joins physically separated platforms, each modeled as a cloud, providing bioinformatics applications that can be used as if they were one single system. The results showed that the storage policy, lowering the time to transfer files used in bioinformatics workflows, really lowered the total time to execute these workflows.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 107–114, 2014.

This paper is organized as follows. Section 2 discusses related work. Section 3 presents the storage policy proposal. Section 4 first presents the BioNimbuZ platform and its services, and after the case study. Finally, our conclusions are presented in Section 5.

2 Related Work

The storage and transfer of files between cloud members of the federation should consider several characteristics in order to perform the operation efficiently. Thus, to increase efficiency, it is essential to analyze the elements that influence the systems response time.

Stockinger and Stockinger [9], in their cost model for data distribution and replication, presented some aspects, among which the following stand out: physical location, bandwidth, size of the file being downloaded and transmission protocol. Although, this work is related to the cost aspects as it has aimed to carry out the operation in the minimum time, which, consequently, would lower the cost.

Another important aspect, which Bermbach et al. [10] highlight, is that dependence on a single-service storage imposes limitations on availability and scalability related to the selection provided, aside from possibly causing delays in executions carried out by the providers of service. To minimize this problem, the authors propose the MetaStorage, a federated storage system using hash tables that can integrate different suppliers through data replication. In this architecture, the agents, who act on the storage providers, perform the storage and retrieve data that are located at the lowest level and provide a generic interface to abstract the technical details of the underlying infrastructure.

Due to the large amount of data, one of the recurring concerns in research involving cloud storage and data compression, Bogdan presents BlobSeer [11], a model for distributed data management, specially implemented to read, write and collate large amounts of information on a large scale, working seamlessly to perform data compression. This approach reduces the disk space needed for storage, and relies on the use of bandwidth to transfer data, however, one should consider the feasibility of this approach due to time and cost required to perform such operations.

3 Storage Policy

The storage policy is a fundamental part of a storage service in federated clouds. Through this service, it is possible to store the files so that they are available when needed.

The storage policy aims to choose the best clouds for storing the files, with the goal of completing the execution of the task in order to lower time. For the proposed policy, some criteria were considered. The weight of each criterion determined according to its importance and impact, based on tests performed in [7], using a simple storage policy. The storage policy provides two functions:

storage output files (upload), containing the result of bioinformatics analysis, and retrieval of input files (download), required to execute the tool to generate the output files.

In the download process (Fig. 1) there are four steps: index calculation and cloud selection, system download, file compression and file transfer. This general overview is detailed in the following subsections.

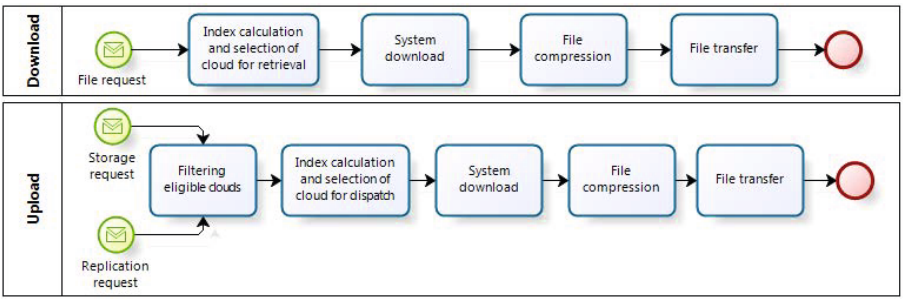


Fig. 1. Download process for storage policy, used when a job request a file and, upload process for storage policy, used to store a file in the federation

The upload Process has the same steps of the download process, with a previous step of filtering eligible clouds. To increase the performance of the storage policy, we propose to replicate files, done with the upload process for copying files to other clouds. This provides availability for the files used in the federated cloud.

Both processes have similar steps, but differ in the implementation phase, filtering the eligibility for receiving the files before selecting one cloud for storing or retrieving a file.

3.1 Filtering Eligible Clouds

This phase is used only in the upload process, since clouds that are candidates to receive one file should have sufficient storage space. The process of filtering eligible clouds aims to select only the ones to be considered for transferring, thus ensuring that one file is not sent to a cloud not able to receive it (due to insufficient space).

3.2 Index Calculation and Cloud Selection

It is necessary to choose the best cloud integrating the federation for one of the operations, download or upload. For this, the clouds are classified according to the following aspects:

– Latency: a measure of the delay of a network, a link or a device to another. Higher latency indicates long delays. Latency can never be entirely eliminated, and it is used as a measure of network performance. Latency varies depending on the load imposed on the network;

- Storage cost: as hybrid architectures enable the existence of various types of clouds, cost of storage is considered for choosing a cloud, thus prioritizing the ones not charging for transferring and data storage;
- Processor cores: a larger number of processor cores in the candidate cloud enables faster execution of the workflow. This is considered by the storage policy, in order to decrease the transfer of files between clouds. It has to be noted that the jobs are executed on machines having the file;
- Workload: if one cloud is overloaded, the ability to perform a certain task as soon as the data are available is low, and may influence the execution time of the entire workflow.

Each aspect is normalized and ranges from 0 to 1. After this, every aspect will be multiplied by its corresponding weight, and sorted in decreasing order by the resulting index. This way, the cloud with the higher index will be chosen for file storage or download, as showed in Equation 1.

$$I = (\frac{l}{max(l)} * wl) + (\frac{sc}{max(sc)} * wsc) + (\frac{wk}{100} * wwk) - (\frac{np}{max(np)} * wnp) \quad (1)$$

In Equation 1: l - latency, sc - storage cost, wk - workload, np number of processors. The weights are: wl - latency, wsc - storage cost, wwk - workload, wnp - number of processors. Each criterion received a weight, according to its impact on transfer time and based on tests detailed on next section. This way, we found: latency 50%, processors 30%, work load 15% and storage cost 5%. The first two criteria were associated the large weights because they are the ones that allow rapid transfer and execution of the work, the focus of this work.

3.3 File Compression

After choosing the cloud, the compression process is performed. This process provides size reduction, and lower file transfer time. Regarding compression, genomic data files, used in our case study in the BioNimbuZ platform, have specific characteristics that allow a high compression ratio. This is due to the fact that files are composed of only four distinct characters (A, C, G, T/U) representing the nucleotides of DNA/RNA, respectively, Adenine, Cytosine, Guanine and Thymine/Uracil, besides spaces. As shown by Rubin [12], the basic approach to text compression is to divide the original text and replace each of these substrings by a code. This is very successfully applied to files with the characteristics mentioned before, where the number of characters is very short and there are many spaces. Thus, the compression of the files strongly reduces its size. The implementation used in this policy is Snappy [13], which prioritizes the speed of a moderate compression.

4 Case Study

Here, we show the results obtained when executing our storage policy in the BioNimbuZ plataform.

4.1 BioNimbuZ Platform

BioNimbuZ platform enables the integration of different cloud computing platforms, meaning that independent, heterogeneous, private or public providers may offer their bioinformatics services in an integrated manner, while maintaining their particular characteristics and internal policies.

BioNimbuZ architecture is composed of services (Monitoring, Security, Discovery, Scheduling, Storage, Fault Tolerance, Job Controller and Service Level-Agreements - SLA). In this paper, we focus on the Storage Service, which decides how to distribute and replicate data among the cloud integrating the federation. For more details about the other services see [14].

In the previous version of BioNimbuZ, the storage policy was implemented in a simple way, selecting the first cloud found. When a job needs a file, it makes a request to the Storage Service, which selects one among the clouds having the file. However, the selected cloud might not be the best. In [7], it was possible to note that at least 19% of the jobs (18 of 96 jobs) spent more than 50% of the time transferring files, while some jobs reached even 79% of the time.

4.2 A Bioinformatics Workflow

The bioinformatics workflow chosen for the case study aims to identify differentially expressed genes in human kidney and liver cancer cells, with fragments sequenced by Illumina sequencers and consists of four phases.

In the first phase (mapping), fragments are mapped to the 24 human chromosomes (1 to 22, X and Y). The goal is to identify the region of the chromosomes where each fragment is located. A set of fragments mapped to the same region allows us to infer that they have the same structural organization of the reference genome. Bowtie [18] was used to do this mapping.

In the second phase, the SAM format output mapping is converted to the BED format with a script for conversion called sam2bed, implemented exclusively for this workflow.

In the third phase, with a script called genome2interval, intervals are generated based on the size of each chromosome. In this phase, from outputs of second and third phases, histograms are generated, which indicate the number of fragments mapped for the range of chromosomes with the suite tool BEDtools [16].

This pipeline has 96 jobs. The files of the 24 chromosomes totalized 2.9GB, the largest, 248MB, and the smallest 51MB. Compressed, the files totalized 893MB, a reduction of 70%.

4.3 Environment and Implementation Details

Each federation used in the tests was composed of three clouds. Each cloud had three nodes, each with Linux Ubuntu 12.04, Apache Hadoop framework [15] and all the tools used in the workflow. The federated cloud was composed of three clouds located at University of Brasilia, Amazon West Coast and Amazon East Coast.

We note that, in clouds, one of the key technologies adopted to execute bioinformatics tools is the Apache Hadoop framework [15], in which the MapReduce model and its distributed file system (HDFS) are used as infrastructure to distribute large scale processing and data storage. This is due to the fact that the parallelization does not require communication among simultaneously processed tasks, since they are independent from each other.

In BioNimbuZ, the index calculation is performed by the Storage Service, which will provide the required information. The workload and processor cores were used to complete operations in the cloud, download the file from Hadoop and compress data, both being affected by the characteristics of cloud computing. The test data, presented in next section, will show the impact of these aspects.

To test the storage policy, two schedulers were used in BioNimbuZ:

- Round Robin (RR), one of the simplest scheduling algorithms for processes in an operating system, assigning time slices to each process in equal portions and in circular order, and handling all processes without priority; and
- Ant Colony Optimization (ACO) [17], a random search algorithm that mimics the behavior of a colony of real ants in search of food, using pheromones to trace the paths, adapting as the executions are carried out.

4.4 Results

We set up four scenarios, joining configuration and scheduler. For each scenario five tests were conducted, totalizing 20 executions, each with 96 jobs. The hardware configuration of cloud nodes in tests were: Type 1, Core(TM)2 Duo CPU E7300, 2GB RAM, 160 GB HD; Type 2 (small), variable processor, 1 core, 650 MB RAM, 20 Gb (HD); and Type 3 (large), Intel Xeon E5-2670, 4 cores, 7,5 GB RAM, 500 GB HD.

- Scenario 1: 1 cloud type 1 and 2 clouds type 2, all running RR Scheduler;
- Scenario 2: 1 cloud type 1 and 2 clouds type 3, all running RR Scheduler;
- Scenario 3: 1 cloud type 1 and 2 clouds type 2, all running ACO Scheduler;
- Scenario 4: 1 cloud type 1 and 2 clouds type 3, all running ACO Scheduler.

Figure 2 shows the execution times in the four scenarios. Our storage policy has shown an average gain of 40%, when compared to the execution time of the original storage policy.

Among the files to be transferred, 50 jobs spent more than 30% of the total time to transfer files, from these, 47 (94%) large files used in the first phase (mapping) had to be transferred. Besides, the transfer time also strongly affected executions in larger machines, 30 files in scenarios using hardware Type 3, compared to the 20 files using hardware Type 2.

Because executions are parallel, it is not possible to guarantee the order, as well as the place of execution (University of Brasilia, Amazon West Coast or Amazon East Coast). Analyzing the greatest transfer time in the log files, we found that these jobs executed at the University of Brasilia, where all the executed jobs showed the same characteristics, low transfer rate and fast execution.

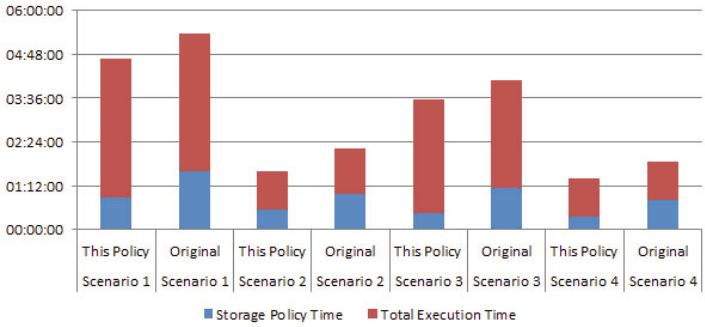


Fig. 2. Workflow runtimes, where the proposal policy has shown an average gain of 40%

5 Conclusion

The storage policy for federated clouds proposed in this paper aims to reduce file transfer time when executing bioinformatics workflows, considering characteristics of the files used in bioinformatics. We developed a case study of our storage policy compared to the original one in BioNimbuZ, a federated cloud designed to execute bioinformatics applications. Results showed that our storage policy performed well when there are clouds with more robust features in the federation, which could be seen by the shorter total execution time. Furthermore, data transfer strongly affected total run time. The file size (chromosome size) did not affect the performance of the storage policy, although it impacts the total execution time. Besides, the run time of jobs were reduced, which contributed to a faster execution of workflow. Compared to other works in the literature, our storage policy was focused on the characteristics of the genomic data files, which were used for improving data transfer.

Despite of the good results obtained by our policy, it could be improved by integrating the storage policy with the Scheduling Service, so that the jobs could choose to send the jobs to the cloud having the files or to the best cloud, aside of requesting the files, as is done now. In addition, integration with SLA allows the user to select features that meet his needs, e.g., as lower costs of storage or data transfer.

References

1. Amazon, Amazon EC2 (July 2014), `http://aws.amazon.com/ec2/`
2. Redkar, T.: Windows Azure Platform, 2nd edn., vol. 1. Apress, Berkeley (2011)
3. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39, 50–55 (2008)
4. Sanderson, D.: Programming Google App Engine: Build and Run Scalable Web Apps on Google's Infrastructure, 1st edn. O'Reilly Media, Inc. (2009)

5. Buyya, R., Ranjan, R., Calheiros, R.N.: InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services. In: Hsu, C.-H., Yang, L.T., Park, J.H., Yeo, S.-S. (eds.) ICA3PP 2010, Part I. LNCS, vol. 6081, pp. 13–31. Springer, Heidelberg (2010)
6. Celesti, A., Tusa, F., Puliafito, A.: How to enhance cloud architectures to enable cross-federation. In: 3rd IEEE Int. Conf. on Cloud Comp., pp. 337–345 (2008)
7. Saldanha, H.V., Ribeiro, E., Holanda, M., Araujo, A., Rodrigues, G., Walter, M.E.M.T., Setubal, J.C., Davila, A.: A cloud architecture for bioinformatics workflows. In: INSTICC, L. (ed.) 1st International Conference on Cloud Computing and Services Science, CLOSER 2011, pp. 1–8 (2011)
8. Lima, D., Moura, B., Oliveira, G., Ribeiro, E., Araujo, A., Holanda, M., Togawa, R., Walter, M.: A Storage Policy for a Hybrid Federated Cloud platform: A Case Study for Bioinformatics. In: 2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGrid (2014)
9. Stockinger, H., et al.: Towards a cost model for distributed and replicated data stores. In: Proc. Nineth Euromicro Workshop on Parallel and Distributed Processing, Wien Univ., Austria, pp. 461–467. IEEE Computer Society (2001)
10. Bermbach, D., Klems, M., Tai, S., Menzel, M.: Metastorage: A federated cloud storage system to manage consistency-latency tradeoffs, in. In: Proceedings of the 2011 IEEE 4th International Conference on Cloud Computing, CLOUD 2011, pp. 452–459. IEEE Computer Society, Washington, DC (2011)
11. Nicolae, B.: High throughput data-compression for cloud storage. In: Hameurlain, A., Morvan, F., Tjoa, A.M. (eds.) Globe 2010. LNCS, vol. 6265, pp. 1–12. Springer, Heidelberg (2010)
12. Rubin, F.: Experiments in text file compression. Commun. ACM 19, 617–623 (1976)
13. Google, Snappy (July 2014), `http://code.google.com/p/snappy`
14. Saldanha, H., Ribeiro, E., Borges, C., Araujo, R.G.A., Holanda, M., Walter, R.T.M.E., Setubal, J.C.: Towards a hybrid federated cloud platform to efficiently execute bioinformatics workflows. In: Intech Bioinformatics, pp. 051–0878 (2012)
15. Apache, Apache Hadoop (July 2014), `http://hadoop.apache.org/`
16. Quinlan, A.R., Hall, I.M.: BEDTools: a flexible suite of utilities for comparing genomic features. Bioinformatics 26, 841–842 (2010)
17. de Oliveira, G., Ribeiro, E., Ferreira, D., Arajo, A., Holanda, M., Walter, M.: Acosched: A scheduling algorithm in a federated cloud infrastructure for bioinformatics applications. In: 2013 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 8–14 (December 2013)
18. Langmead, B., Trapnell, C., Pop, M., Salzberg, S.: Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10(3), R25 (2009)

Genome-Wide Identification of Non-coding RNAs in *Komagatella pastoris* str. GS115

Hugo Schneider[1], Sebastian Bartschat[2], Gero Doose[2], Lucas Maciel[1], Erick Pizani[1], Marcelo Bassani[1], Fernando Araripe Torres[3], Sebastian Will[2], Tainá Raiol[3], Marcelo Brígido[3], Maria Emília Walter[1], and Peter Stadler[2,3,4,5,6,7,8]

[1] Department of Computer Science, Institute of Exact Sciences, University of Brasília, 70910-900, Brasília, DF, Brazil
[2] Bioinformatics Group, Department of Computer Science, and Interdisciplinary Center for Bioinformatics, University of Leipzig, Härtelstraße 16-18, D-04107 Leipzig, Germany
[3] Department of Cellular Biology, Institute of Biology, University of Brasília, 70910-900, Brasília, DF, Brazil
[4] Max Planck Institute for Mathematics in the Sciences, Inselstraße 22, D-04103 Leipzig, Germany
[5] Fraunhofer Institut für Zelltherapie und Immunologie, IZI Perlickstraße 1, D-04103 Leipzig, Germany
[6] Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, A-1090 Wien, Austria
[7] Center for non-coding RNA in Technology and Health, University of Copenhagen, Grønnegårdsvej 3, DK-1870 Frederiksberg C, Denmark
[8] Santa Fe Institute, 1399 Hyde Park Rd., Santa Fe, NM 87501

Abstract. The methylotrophic yeast *Komagatella pastoris* is a relevant bioengineering platform for protein synthesis. Even though non-coding RNAs are well known to be key players in the control of gene expression no comprehensive annotation of non-coding RNAs has been reported for this species. We combine here published RNA-seq data with a wide array of homology based annotation tools and *de novo* gene predictions to compile the non-coding RNAs in *K. pastoris*.

Keywords: Non-coding RNA, Transcriptome, *Komagatella pastoris*, *Pichia pastoris*, snoRNA, long ncRNA.

1 Introduction

Non-coding RNA (ncRNA) is a class of RNA molecules that does not harbor a functional open reading frame (ORF), and thus does not code for protein. Besides the well-known ribosomal RNA (rRNA) and transfer RNA (tRNA), many different types of ncRNA have been recently described. These molecules were found to be involved in many aspects of cell functioning, including in the regulation of gene expression and differentiation. Small and long ncRNA were

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 115–122, 2014.

described in all domains of life, especially among eukarya, where the amount of ncRNA has been correlated to organismal complexity [14]. Although among metazoans ncRNAs seem to have a fundamental role in chromatin organization and RNA metabolism, in the fungal kingdom there is only limited information about their roles in the cell's informational metabolism.

The fungal species that is most extensively studied also from the point of view of ncRNAs is *Saccharomyces cerevisiae*. It harbors a wide variety of long regulatory RNAs, reviewed in [23]. Bakers yeast is a quite atypical system, however, in particular with regard to its ncRNAs. It has recently lost the RNA interference (RNAi) pathway [8], and processes many of its Small Nucleolar RNA (snoRNAs) in an atypical manner in conjunction with a dramatic, genome-wide loss of introns [15]. MicroRNA-like RNAs (milRNAs) and Dicer-independent siRNA-like RNAs were discovered by small RNA-seq in the Neurospora genus [11] and in several other diverse fungal species including *Metarhizium anisopliae* [26] and *Sclerotinia sclerotiorum* [25]. These ncRNAs are probably produced by different biogenesis pathways, contrasting them with the microRNAs of plants and animals.

The methylotrophic yeast *Komagatella pastoris* (formerly *Pichia pastoris*) is widely used as a bioengineering platform for producing industrial and biopharmaceutical proteins and its genome and transcriptome starts to be studied [5,7] to improve its annotation and thus its biotechnological skills. Here we report on an *in silico* search for ncRNAs in the genome of *K. pastoris* GS115, based on the analysis of its genome and its transcriptionally active regions.

2 Materials and Methods

Homology-Based Annotation. The *K. pastoris* str. GS115 [6] genome sequence, comprising four chromosomes and three additional contigs of unplaced sequences, accessions FN392319-FN392325, were downloaded from the NCBI Genome Data-base. An initial set of candidate RNAs was determined using `Infernal` [16], a software for ncRNA detection based on the collection of covariance models provided by the Rfam database 11.0 [4]. In addition, we executed `tRNAscan-SE` to identify the complement of tRNAs.

Identification of Small Nucleolar RNAs. In addition to the `Rfam`-based search we employed snoStrip [2] using known fungal snoRNAs as a starting data set. Furthermore, we checked whether predicted snoRNAs were located within a known transcript or transcribed regions according to gene models derived by Cufflinks [20].

Identification of Long ncRNAs. We used the four non-strand-specific Illumina RNA-seq data sets SRX109531 to SRX109534 comprising poly-A selected RNA of *K. pastoris* grown in media containing either methanol or glycerol [13]. The data were mapped twice to the *K. pastoris* genome using `Segemehl` [10], once allowing unsplit reads only (used for differential expression and ncRNA detection) and once enabling mapping with splits, including non-colinear splits.

The mapping results were input to `Cufflinks` [20] version 2.0.2 to detect transcripts. In addition we determined "coverage loci", defined as contiguous regions with a minimum coverage of 8. "Coverage loci" separated by less than 50 nt were merged, loci with less than 50 nt of total length were removed.

Expression Levels and Differential Expression of lncRNAs. "Coverage loci" regions were determined based on RPKM (Reads Per Kilobase of gene per Million of reads) values computed as RPKM $= 10^9 C/(NL)$, where C is the number of reads mapped onto the locus, N is the total number of mappable reads in the experiment, and L is the total length of the locus in base pairs. The coordinates of the "coverage loci" were compared to *K. pastoris* GS115 annotation to identify overlapping regions. All "coverage loci" overlapping with annotated coding genes were removed. The remaining transcripts were further filtered using `Blastx` against non redundant proteins of GenBank, with e-value cutoff $< 10^{-20}$ and a minimum coverage of 50%. Differential expression of long ncRNAs between methanol and glycerol media was quantified with `Cuffdiff` [19].

Splice Junctions and Atypical Transcripts. Splice junctions were determined by `Haarz`, a component of the `Segemehl` suite. In order to reduce the chance of mapping artifacts splice junctions not matching certain mapping criteria or not supported by at least three split reads were filtered out. Only reads that entirely mapped to the four chromosomes were retained, i.e., trans-spliced to the rRNA locus or one of two short contigs were ignored. Canonical (GT:AG) splice junctions were used to determine the reading direction of the transcripts containing them. These split-read mapping was used to identify non-standard transcripts and to investigate snoRNA host genes.

Sequence Conservation. Multiple sequence alignments were computed with the `multiz/TBA` pipeline [3] spanning the entire phylogenetic range of the hemiascomycote, including *Saccharomyces cerevisae* and *Yarrowia lipolytica*. An `RNAz` screen to identify putative conserved ncRNAs was performed as described in [9].

3 Results and Discussion

3.1 Small Structured ncRNAs

The initial `Infernal`-based search resulted in 376 ncRNA candidates, of which most were tRNAs or snoRNAs. After resolving conflicting matches of multiple covariance models to the same genomic location and removing obvious false positives, we retained 186 ncRNA homologues of others known in Rfam 11.0 (in our annotation), confirmed with other information obtained from homology-based annotation.

A `tRNAscan-SE` search resulted in 123 tRNAs, 29 contained an intron. These tRNAs comprise 44 distinct anti-codons encoding all the 20 amino acids. As expected, there is no trace of a minor spliceosome and no *bona fide* tRNASec, since fungi, like higher plants, lack a selenocystein insertion system [24]. As described in [6], the rRNA operon is contained in an extra contig (FN392325)

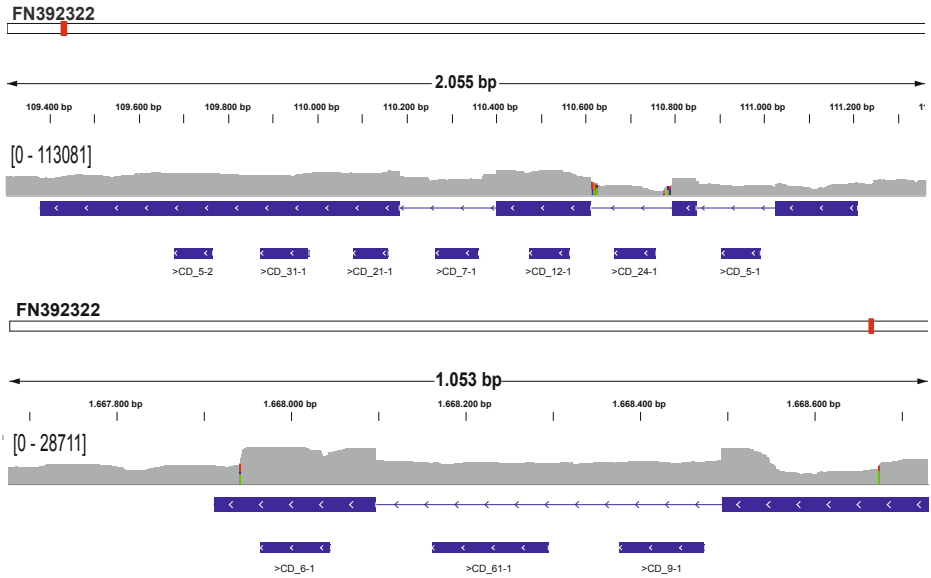


Fig. 1. RNA-seq density and inferred transcript structures of two snoRNA clusters

in the genome assembly. We found 21 copies of 5S rRNA, distributed across all the chromosomes. Apart from tRNAs and rRNAs, we detected 46 structured ncRNAs.

As in other Saccharomycotina we obtained one copy for each of the five snRNAs of the major spliceosome (U1, U2, U4, U5 and U6). As for other fungi [18] there is no trace of minor spliceosomal snRNAs. We found one copy each of the RNA components of the signal recognition particle, and of the RNAses P and MRP. We were unable, however, to detect the RNA component of telomerase. This is not unexpected, given the rapid evolution of this RNA family [17].

Riboswitches are widely found in bacteria, but are rare in animals and fungi. A notable exception is the thiamine sensing TPP riboswitch, which we found close to the 5' end of the *K. pastoris* Thi4 homologue, the usual location where these switches have been found in several ascomycetes [12].

Of the snoRNA candidates predicted by `Infernal` (39) and `snoStrip` (47), 30 were found by both methods. Using the split transcript model, we could place 31 of the `Infernal` predicted hits on transcriptionally active loci, but two box H/ACA and two box C/D were found in the wrong direction in relation to transcript orientation. Since the orientation of Cufflinks produced models is biased by the presence of donor and acceptor splicing sites, these four predict boxes could still be functional snoRNAs. In total we had found five box H/ACA, two intronic and three exonic, and 23 box C/D, nine in introns and two in exons. Furthermore, we identified a single copy of the U3 snoRNA.

A peculiar feature of Hemiascomycetes is a dramatic loss of introns which is accompanied by a change in the processing of snoRNAs [15]. While snoRNAs

are ancestrally processed from introns, they are mostly found in exonic positions in *Kluyveromyces* and *Saccharomyces*. In *K. pastoris* we found an intermediate stage that is well illustrated by the two snoRNA clusters shown in Figure 1. All the 31 snoRNAs are located within expressed loci according to the available RNA-seq data. Only 13 are located in introns of the annotated protein-coding genes, while ten are found in exonic positions of coding genes. The remaining eight are placed in non-coding host genes. Four of them are intronic. Twelve box C/D snoRNAs were found in three different box C/D clusters. The major cluster co-localizes to a long ncRNA located in the chromosome 4 (discussed below). This RNA harbors three introns, each one of them containing one box C/D. Four additional box C/D snoRNAs were found in the last two exons of the transcript (Figure 1).

3.2 Long Non-coding RNAs

The merged transcripts of both growing conditions cover roughly half of the entire genome and define 3,542 "coverage loci", of which 2,058 completely overlap with protein coding or annotated ncRNA. Only 136 "coverage loci" not overlapping at least partially were further filtered to remove potential ORF coding transcript not annotated in the *K. pastoris* GS115 project. We discarded 86 "coverage loci" due to sequence similarity with coding sequences from other fungi, 85 of them perfect hits to the CDS annotation of *K. pastoris* strain CBS7435, highlighting the incompleteness of the *K. pastoris* GS115 annotation. The remaining 50 loci were considered to be intergenic lncRNAs. Non coding character of most of them (62%) was supported by an independent predictor [1]. Their length ranges from 122 to 1,912 bp, with an average size of 593 bp. We found 94% longer then 200 nt, the minimum length of "lincRNAs", according to the literature.

In the yeast *S. cerevisiae* a large number of anti-sense RNAs (asRNAs) have been reported to control gene expression, among a large class of so-called XUT RNAs [22]. Although most of them are unstable and not likely to be included in a poly-A selected library, we nevertheless investigated the available evidence for asRNAs. The identification of asRNA is very difficult from non-strand-specific sequencing data and is essentially restricted to spliced RNAs. Due to the relatively low abundanced of spliced transcripts it is not surprising that only six antisense transcripts could be unambiguously identified.

Only a tiny fraction (0.15%) of the mapped reads were mapped across splice junctions, validating more than 550 splice junctions within the existing annotation and identifying 118 novel ones. Together, these account for more then 70% of the split reads. The remainder corresponds to candidates for circular RNAs, local strand-switching transcripts, and long-range trans-splicing (Table 1).

3.3 Differentially Expressed lncRNA

In order to determine whether lncRNAs are involved in the medium-specific gene expression patterns we investigated expression changes using the available

Table 1. Relation between annotation and high confidence splice junctions in different splicing types

Splicing type	Splice junctions (read support)					
	all		intergenic		%	
normal	586	(295,299)	150	(38,392)	25.6	(13.0)
circular	1,342	(8,139)	94	(753)	7.0	(9.3)
strand-switch	11,627	(60,753)	961	(5033)	8.3	(8.3)
trans	61	(415)	14	(36)	23.5	(8.7)

Table 2. Differentially expressed long ncRNAs

chr	start	fold change	p-value	precursor RNA	mature RNA	exons	comment
lincRNA							
chr1	1,852,350	3.38793	9.50E-04	190	190	1	
chr1	1,852,845	2.50889	5.00E-05	153	153	1	Possible 5'UTR
chr1	2,797,859	5.91132	2.70E-03	632	591	3	telomeric
chr2	2,347,935	-2.12367	5.00E-05	678	663	2	3' to Py carboxilase
chr2	2,355,594	2.41426	1.25E-03	1,649	1,649	1	
chr3	384,136	>10	5.00E-05	212	212	1	
chr4	109,379	1.4597	2.70E-03	1,829	1251	4	snoRNA cluster
antisense RNA							
chr4	34,307	2.65094	7.00E-04	2,398	2219	3	Floc9-AS
chr4	186,332	2.24018	5.00E-05	3,750	3536	2	Sec16-AS
chr4	372,117	2.69379	2.00E-04	2,320	2219	2	Aim1-AS

RNA-seq data. To this end, we used the reads mapped in non-split mode overlapping the annotated features. From the observed 3,542 "coverage loci", most of them overlap one or, in fewer cases, more than one protein coding genes, or annotated ncRNAs. A total of 322 "coverage loci" showed significant differences between the two growth conditions. Of these, 152 were upregulated and 170 downregulated in the presence of methanol. This set contained seven lincRNAs and three asRNAs listed in Table 2. Most of them are repressed in the presence of methanol while a single one was shown to increase expression in this situation. Interestingly, this lincRNA is located nearby to a Pyruvate Carboxylase gene, that is shown to be a key controler of energetic metabolism in *K. pastoris* [21].

One of the differentially expressed genes is a 1,829 nt primary transcript carrying a cluster of snoRNA as discussed above. It is processed to a 1,251 nt mature transcript. Its function beyond a template for snoRNA is not known, and it may serve solely as host gene for the cluster of snoRNA. This transcript carries the largest cluster of snoRNA and its induction in the yeast in the presence of glycerol, but not methanol, suggest it may be of importance for maintaining a specific physiological state.

4 Concluding Remarks

Non-coding RNAs have been identified as important players in the gene expression control of virtually all organisms studied to-date at any detail. Nevertheless, a comprehensive annotation of non-coding RNAs is usually not part

of initial genome annotation efforts in newly sequenced organisms. This fact appears to be a consequence of the lack of a simple toolkit for this purpose. The diversity of ncRNA classes in terms of their size, conservation, and patterns of molecular evolution requires the combination of wide variety of computational tools for homology-based search, *de novo* RNA gene finding, as well as the integration of experimental RNA-seq data. Here, we have combined to available evidence to annotate all major classes of small ncRNAs and at least the most prevalently expressed long non-coding RNAs.

Despite the practical importance of *K. pastoris* the available information is far from complete. As available RNA-seq data are not strand-specific, only a handful of antisense RNAs could be identified. On the other hand, small structured RNAs have not been annotated in a systematic manner in Ascomycota beyond the *Saccharomyces* and *Aspergillus* clades. This contribution thus provides an important corner stone for subsequent annotation efforts aiming at yeast ncRNAs.

The presence of ncRNAs that are differentially expressed in response to changes in the main carbon source provided by the growth medium hints at the importance of non-coding RNAs in the regulation of *K. pastoris*' fundamental biochemical processes. As such these RNAs become interesting targets for functional characterization and, eventually, for innovations in biopharmaceutical production.

References

1. Arrial, R.T., Togawa, R.C., Brigido, M.M.: Screening non-coding RNAs in transcriptomes from neglected species using PORTRAIT: case study of the pathogenic fungus Paracoccidioides brasiliensis. BMC Bioinform. 10(1), 239 (2009)
2. Bartschat, S., Kehr, S., Tafer, H., Stadler, P.F., Hertel, J.: snoStrip: A snoRNA annotation pipeline. Bioinformatics, btt604 (2013)
3. Blanchette, M., Kent, W.J., Riemer, C., Elnitski, L., Smit, A.F.A., Roskin, K.M., Baertsch, R., Rosenbloom, K., Clawson, H., Green, E.D., Haussler, D., Miller, W.: Aligning multiple genomic sequences with the threaded blockset aligner. Genome Res. 14, 708–715 (2004)
4. Burge, S.W., Daub, J., Eberhardt, R., Tate, J., Barquist, L., Nawrocki, E.P., Eddy, S.R., Gardner, P.P., Bateman, A.: Rfam 11.0: 10 years of RNA families. Nucleic Acids Research, gks1005 (2012)
5. Cregg, J.M., Tolstorukov, I., Kusari, A., Sunga, J., Madden, K., Chappell, T.: Expression in the yeast *Pichia pastoris*. Methods Enzymol. 463, 169–189 (2009)
6. De Schutter, K., Lin, Y.-C., Tiels, P., Van Hecke, A., Glinka, S., Weber-Lehmann, J., Rouzé, P., Van de Peer, Y., Callewaert, N.: Genome sequence of the recombinant protein production host Pichia pastoris. Nature Biotechnol. 27(6), 561–566 (2009)
7. Dikicioglu, D., Wood, V., Rutherford, K.M., McDowall, M.D., Oliver, S.G.: Improving functional annotation for industrial microbes: a case study with Pichia pastoris. Trends in Biotechnology 32(8), 396–399 (2014)
8. Drinnenberg, I.A., Weinberg, D.E., Xie, K.T., Mower, J.P., Wolfe, K.H., Fink, G.R., Bartel, D.P.: RNAi in budding yeast. Science 326(5952), 544–550 (2009)
9. Gruber, A.R., Findeiß, S., Washietl, S., Hofacker, I.L., Stadler, P.F.: RNAz 2.0: improved noncoding RNA detection. Pac. Symp. Biocomput. 15, 69–79 (2010)

10. Hoffmann, S., Otto, C., Doose, G., Tanzer, A., Langenberger, D., Christ, S., Kunz, M., Holdt, L.M., Teupser, D., Hackermüller, J., Stadler, P.F.: A multi-split mapping algorithm for circular RNA, splicing, trans-splicing, and fusion detection. Genome Biology 15, R34 (2014)
11. Lee, H.-C., Li, L., Gu, W., Xue, Z., Crosthwaite, S.K., Pertsemlidis, A., Lewis, Z.A., Freitag, M., Selker, E.U., Mello, C.C., et al.: Diverse pathways generate microRNA-like RNAs and dicer-independent small interfering RNAs in fungi. Molecular Cell 38(6), 803–814 (2010)
12. Li, S., Breaker, R.R.: Eukaryotic TPP riboswitch regulation of alternative splicing involving long-distance base pairing. Nucl. Acids Res. 41, 3022–3031 (2013)
13. Liang, S., Wang, B., Pan, L., Ye, Y., He, M., Han, S., Zheng, S., Wang, X., Lin, Y.: Comprehensive structural annotation of Pichia pastoris transcriptome and the response to various carbon sources using deep paired-end RNA sequencing. BMC Genomics 13(1), 738 (2012)
14. Mattick, J.S., Makunin, I.V.: Non-coding RNA. Human Molecular Genetics 15(suppl. 1), R17–R29 (2006)
15. Mitrovich, Q.M., Tuch, B.B., Francisco, M., Guthrie, C., Johnson, A.D.: Evolution of yeast noncoding RNAs reveals an alternative mechanism for widespread intron loss. Science 330(6005), 838–841 (2010)
16. Nawrocki, E.P., Kolbe, D.L., Eddy, S.R.: Infernal 1.0: Inference of RNA alignments. Bioinformatics 25(10), 1335–1337 (2009)
17. Qi, X., Li, Y., Honda, S., Hoffmann, S., Marz, M., Mosig, A., Podlevsky, J.D., Stadler, P.F., Selker, E.U., Chen, J.: The common ancestral core of vertebrate and fungal telomerase RNAs. Nucleic Acids Research, gks980 (2012)
18. Russel, A.G., Charette, M., Spencer, D.F., Gray, M.W.: An early evolutionary origin for the minor spliceosome. Nature 443, 863–866 (2006)
19. Trapnell, C., Hendrickson, D.G., Sauvageau, M., Goff, L., Rinn, J.L., Pachter, L.: Differential analysis of gene regulation at transcript resolution with RNA-seq. Nature Biotech. 31 (2013)
20. Trapnell, C., Williams, B.A., Pertea, G., Mortazavi, A., Kwan, G., van Baren, M.J., Salzberg, S.L., Wold, B.J., Pachter, L.: Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. Nat. Biotechnol. 28, 511–515 (2010)
21. Unrean, P.: Pathway analysis of Pichia pastoris to elucidate methanol metabolism and its regulation for production of recombinant proteins. Biotechnology Progress 30(1), 28–37 (2014)
22. van Dijk, E.L., Chen, C.L., d'Aubenton Carafa, Y., Gourvennec, S., Kwapisz, M., Roche, V., Bertrand, C., Silvain, M., Legoix-Né, P., Loeillet, S., Nicolas, A., Thermes, C., Morillon, A.: XUTs are a class of Xrn1-sensitive antisense regulatory non-coding RNA in yeast. Nature 475, 114–117 (2011)
23. Wu, J., Delneri, D., O'Keefe, R.T.: Non-coding RNAs in Saccharomyces cerevisiae: what is the function? Biochemical Society Trans. 40(4), 907–911 (2012)
24. Xu, X.-M., Carlson, B.A., Mix, H., Zhang, Y., Saira, K., Glass, R.S., Berry, M.J., Gladyshev, V.N., Hatfield, D.L.: Biosynthesis of selenocysteine on its tRNA in eukaryotes. PLoS Biology 5(1), e4 (2006)
25. Zhou, J., Fu, Y., Xie, J., Li, B., Jiang, D., Li, G., Cheng, J.: Identification of microRNA-like RNAs in a plant pathogenic fungus Sclerotinia sclerotiorum by high-throughput sequencing. Molec. Genetics and Genomics 287(4), 275–282 (2012)
26. Zhou, Q., Wang, Z., Zhang, J., Meng, H., Huang, B.: Genome-wide identification and profiling of microRNA-like RNAs from Metarhizium anisopliae during development. Fungal Biology 116(11), 1156–1162 (2012)

Multi-scale Simulation of T Helper Lymphocyte Differentiation

P. Tieri[1], V. Prana[1], T. Colombo[1], D. Santoni[2], and F. Castiglione[1]

[1] CNR Consiglio Nazionale delle Ricerche,
IAC Istituto per le Applicazioni del Calcolo, Roma, Italy
{p.tieri,v.prana,t.colombo,f.castiglione}@iac.cnr.it
[2] CNR Consiglio Nazionale delle Ricerche,
IASI Istituto di Analisi dei Sistemi e Informatica, Roma, Italy
daniele.santoni@iasi.cnr.it

Abstract. The complex differentiation process of the CD4+ T helper lymphocytes shapes the form and the range of the immune response to different antigenic challenges. Few mathematical and computational models have addressed this key phenomenon. We here present a multiscale approach in which two different levels of description, i.e. a gene regulatory network model and an agent-based simulator for cell population dynamics, are integrated into a single immune system model. We illustrate how such model integration allows bridging a gap between gene level information and cell level population, and how the model is able to describe a coherent immunological behaviour when challenged with different stimuli.

Keywords: Computational immunology, T helper lymphocyte, CD4+ T cell differentiation, gene regulatory networks, immunoinformatics, CD4+ T cell dogma.

1 Introduction

The immune system provides defence against environmental pathogens and foreign antigens to the host organism. To accomplish this mission, the mammalian immune system orchestrates a variety of highly specialized immune cell subpopulations, including both pathogenic effector cells and protective regulatory cells. This complex ensemble of immune cells branches into two main divisions, the innate and adaptive immunity compartment [1]. Maintenance of proper balance of the different subpopulations throughout homeostasis and immune responses relies on extensive communication among cellular actors, mediated by cytokines (a.k.a. interleukins), i.e., secreted small molecules that modulate abundance and function of the different immune cells. In this variegated landscape, the naïve and activated CD4+ "helper" T lymphocytes contribute as key subset in regulating and supporting immune responses. In response to specific cues from specialized antigen-presenting cells (APCs) and cytokines, naïve CD4+ T cells may be induced towards four best-characterized routes of differentiation in terms of patterns of cytokine production and immune function: effector T helper

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 123–134, 2014.

type 1 (Th1), type 2 (Th2) and type 17 (Th17) cells, and regulatory T (Treg) cells [2]. Understanding the dynamics of the activation and differentiation of such lymphocytes plays a central role in providing potential therapeutic interventions for a number of immune-related dysfunctions and diseases. In this perspective, a number of modelling approaches to study CD4+ T cell differentiation have been proposed, attempting to explain, just to cite a few, Th1/2 lineage choice and maintenance [3]; complex influence of transcription factor dynamics [4]; the expression of, and interactions between the master regulators determining the reciprocal phenotypic polarization between Th17 and induced Treg repertoires [5]; the differentiation into Th1, Th2, Th17 and induced Treg using an integrated systemic approach [6] and Boolean network models [7], among others. Such approaches rely on different modelling strategies and techniques, encompassing ordinary and partial differential equations (ODEs, PDEs), Boolean networks, agent-based models (ABMs). A comprehensive review has been recently published by Carbo et al. [8]. Here we describe our own implementation of a multiscale computational model for CD4+ T helper cell differentiation, which combines two levels of description -a genetic regulation dynamics and a cell-level population dynamics- into a unified, comprehensive simulation system of the immune response. The aim of this integrative work is to construct a flexible and detailed model of the immune response, able to show differentiated responses to distinct exogenous stimuli and, at the same time, to construct a suitable computational environment in which network models of Th differentiation can be tested for consistency.

Naïve CD4+ T cells can differentiate into both effector cells (such as Th1, Th2 and Th17) [9, 10] and protective regulatory cells (Treg) [11], according to the pathogenic stimulus encountered. In the acute phase response (i.e., the immune phase response characterized by fever, changes in vascular permeability and in the biosynthetic, metabolic and catabolic profiles of many organs that initiates immediately after an infection or physical trauma has occurred), activated CD4+ T lymphocytes serve a key role of junction between innate and adaptive immune system by secreting specific arrays of cytokines and chemokines that, respectively, recruit and activate other leukocytes to attack and destroy the pathogens as well as to repair the tissue damage. At later stages, the suppressive intervention of regulatory T cells becomes crucial for recovery from the acute phase response [12, 13]. Broadly, each T helper cell fate is chiefly established by cytokine-mediated up-regulation of lineage-specific transcription factors and reinforced by positive feed-back loops as well as negative regulation of the expression or function of transcription factors of other lineages [2]. Upon the interaction of the invading pathogen with pattern recognition receptors (PPRs) at the surface of innate immune cells such as neutrophils, macrophages, natural killer cells, dendritic cells, the latter start to produce various inflammatory cytokines triggering the appropriate T helper cell response. Viruses and intracellular bacteria cause innate immune cells to produce interferons-alpha and -beta (INF-α and -β) and the interleukin (IL)-12. These stimuli induce the expression of the Tbet transcription factor in naïve T cells, a master regulator of the Th1-polarizing paradigm, and result in the proliferation of IFN-γ -producing Th1 cells [14]. Helminth parasite infection promotes secretion of IL-4, IL-5 and IL-13 by innate cells, that skews undifferentiated Th0 cells to express the Th2-response master regulator GATA3 and to amplify secretion of these cardinal Th2 cytokines (i.e., IL-4, IL-5 and IL-13) [15]. Fungal and bacterial infections drive secretion of TGF-β and IL-6 by innate cells, that directs the

differentiation program of Th17 lineage by inducing its essential transcriptional regulator ROR-γt and thus production of its signature cytokine IL-17 [16]. Finally, Treg cells subset develops from naïve Th0 cells in response to TGF-β in the absence of further pro-inflammatory cues to avert the danger of exaggerated immune responses. Differentiation of such CD4+ T lineage critically depends on activation of the transcription factor Foxp3, and it is functionally characterized by the production of suppressive cytokines such as IL-10, TGF-β or IL-35 [12]. Taken together, these pictures provide a glimpse of the complications involved in the differentiation process but, at the same time, provide suitable information for the construction of a minimalistic model of Th differentiation and activation. In particular, with the present work we focus on the relationship among the intracellular gene activation level that drives the Th differentiation and the *mesoscopic intercellular* signalling scale that describes the cell population dynamics. We have integrated two levels of description and related existing models: the gene activation/inhibition relationship level, identified in a Boolean gene-regulatory network [17], and the intercellular immune cell cooperation level, modelled following the agent-based paradigm [18], as described in the next section.

2 Methods

The computational model of the immune system that we employ [18] has been conceived to allow the dynamic representation of immunological hypotheses and their preliminary testing. From the technical viewpoint, the model follows the agent-based paradigm in which all cellular entities are individually represented. This modelling paradigm has its strength in flexibility and relative easiness of implementation of various conjectures and ideas about how cell interact and cooperate to mount an effective immune response. Agent-based models are based on the paradigm for complex systems inspired by Von Neumann's cellular automata (CAs) [19]. Likewise CAs, it consists of discrete time and space parameterization, and discrete entities, or agents, representing relevant cells and molecules equipped with specific receptors and functional capabilities, which reflect experimental observations. The model incorporates several immunological working assumptions: i) the clonal selection theory of Burnet [20]; ii) the clonal deletion theory (i.e., thymus education of T lymphocytes) [21]; iii) the hypermutation of antibodies [22]; iv) the replicative senescence of T-cells, or the Hayflick limit (i.e., a limit in the number of cell divisions) [23]; v) T-cell anergy [24] and Ag-dose induced tolerance in B-cells [25]; vi) the danger theory [26, 27]; vii) the idiotypic network theory [28]; vii) the attrition conjecture [29, 30]. Variations of the core-basic model have been used to simulate different phenomena ranging from viral infection (e.g., Human Immunodeficiency Virus, Epstein-Barr Virus) [31, 32] to cancer immunoprevention and type I hypersensitivity [33, 34]. Most of these can be toggled on and off. The immunological entities include the major classes of cells of the lymphoid lineage (T helper lymphocyte subsets, or Th, cytotoxic T lymphocytes or Tc, B lymphocytes, antibody-producer plasma cells, or PLB, and natural killer cells, NK) and some of the myeloid lineage, i.e., macrophages (MA, subdivided in MA1 and MA2), and dendritic cells (DCs). Helper T cells are separated in Th0 (naïve), Th1, Th2, Th17 and Tregs. B and plasma cells are also divided into two phenotypes, B-1 and B-2 according to the antibody isotype IgM or IgG they respectively produce.

All these entities interact with each other following a set of rules describing the different phases of the recognition and response of the immune system against a pathogen. In particular, the model takes into account phagocytosis, antigen presentation, cytokine release, cell activation from inactive or anergic states to active states, cytotoxicity, and antibody secretion. Cells communicate through receptor binding and cytokines. Entities live and behave according to algorithmic rules coding for established or conjectured immunological knowledge. A simplified rule for Th differentiation would account for the dominance of key cytokine in the surrounding of a Th0 to determine its maturation fate. In the present work instead we pinpoint this specific rule and expanded its level of sophistication by substituting it with a simplified, yet well sorted, gene relationship network model taken from the work of Mendoza et al. [17]. The regulatory circuitry of Th-switch has been studied in several works aimed to investigate the mechanisms regulating the balance of different kinds of Th cells. In 2006 Mendoza [35] proposed a network to model the switch of Th cells into Th1 and Th2 . In recent years new kinds of Th subtypes were discovered and new models were proposed [36, 37] in order to include Th17 and Treg fate. Several other genes/proteins were included in the network and a system modelled with differential equations was designed to find stable configurations characterizing the Th differentiation. Four single attractors were identified (besides the trivial state where all nodes are inactivated, mapped to Th0) representing Th1, Th2 Th17 and Treg.

There have been previous attempts to integrate in a multiscale approach a cellular agent-based model with its detailed intra-cellular dynamics of gene regulatory network (GRN) of CD4+ T differentiation [38]. The present work expands above approach to include not only Th1 and Th2 but also Th17 and Treg fates in the differentiation model as in [37].

Figure 1 schematizes the gene regulatory network from [37]. The network has been organised according to the different functional compartments of the cell. The differentiation fate of a simulated CD4+ T cell depends on the input stimuli sensed by its membrane receptors, in particular, by the TCR (T-cell receptor, able to bind antigens presented by MHC-II complex on APCs) and by various cytokine receptors (i.e., receptors for IL-6, IL-23, IL-10, TGF-b, IL-2, IL-12, IL-18, IL-4, IFN-β and IFN-γ). Upon the activation of these network nodes, the activation level of key transcription factors and genes ultimately leading to the production of Th cell subset hallmark cytokines (i.e., IL-10, IL-17, IL-6, IFN-g and IL-4) is computed. Every node can assume a binary deactivated (0) or activated (1) state, and is linked to a set of experimentally associated nodes, which can contribute to activate (continuous black line) or to inhibit (dotted red line), as reported in fig. 1. The global network state is synchronously updated according to a common rule used for updating single nodes. A node is activated (i.e., turning its state to 1 at time step $t + 1$) if and only if at time t there is at least one node in the set of its activators are turned on (state 1) and all nodes in its inhibitory set are turned off (state 0); otherwise, its state is set to 0 (deactivated). This is a typically used simplification of a more complex realistic situation but represent the only viable modelling choice in absence of knowledge about the activation of each single gene. The Boolean updating function is formally defined as follows:

$$x_{i,t+1} = \left(\vee_{j\in A_i} x_{j,t}\right) \wedge \left(\neg \wedge_{j\in I_i} x_{j,t}\right)$$

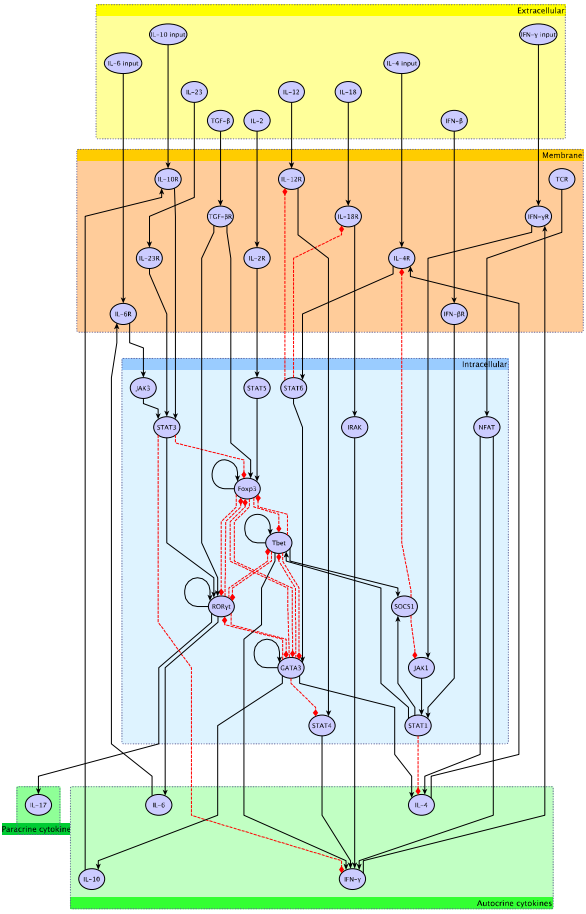


Fig. 1. Scheme of the gene regulatory network of Th differentiation adapted from [37]. Black lines: activation; red lines: repression/deactivation. Depending on the input stimuli that engage cell surface receptors (namely, antigen presenting MHC-peptide complex on TCR, IL-6, IL-23, IL-10, TGF-b, IL-2, IL-12, IL-18, IL-4, IFN-b and IFN-g), the network computes the activation level of key transcription factors for each cell fate (Th1, Th2, Th17 and Treg) and the expression level of genes resulting in the production of signature cytokines such as IL-10, IL-17, IL-6, IFN-g and IL-4.

where we use the notation $x_{i,t}$ to define the state of node i at time t, A_i is the set of nodes activating node i, and I_i is the set of nodes inhibiting node i. For simplification, all cells release the same amount (indicated by ω) of cytokines. This implies that all cytokines have the same efficacy in exercising their action. At each macro time step every Th0 cell 'senses' the concentration of the input cytokines in the same lattice point and sets the activation level of the corresponding input nodes. The activation is modelled as a stochastic event with probability p given by a sigmoid function depending on the concentration c of the input cytokine:

$$p = c^2/(\omega^2 + c^2).$$

As shown in the fig. 1 the network is made of 40 nodes and 67 edges. In order to distinguish the input nodes from the internal ones (that have a feedback effect) we have added four nodes from the original network, namely, 'IFN-γ input', 'IL-4 input', 'IL-6 input' and 'IL-10 input'. We then performed the dynamical simulation according to the rule reported above, until a fixed point is reached (this requires not more that 20 iterations). Four different fixed points are reported in [37] identifying the four Th subtypes: Th1, Th2 Th17 and Treg (Th0 nodes remain Th0 if no input is activated). These are characterized by the activation of a set of signature genes as follows:

- Th1: IFN-γ, IFN- γR, SOCS1, TBET
- Th2: GATA3, IL-10, IL-10R, IL-4, IL-4R, STAT3, STAT6
- Th17: IL-17, IL-6, IL-6R, JAK3, ROR–γt, STAT3
- Treg: FOXP3

According to the final network configuration, and after the discrete dynamics is applied, the transition to the new Th phenotype is operated. The phenotype differences are mirrored by a different pattern of secreted cytokines [39] which greatly influences the overall immune response dynamics.

3 Results and Discussion

When challenged with different stimuli, the above multiscale model provides a dynamical, time-dependent output depicting the character of the mounted adaptive immune response in terms of relative predominance of one T helper phenotype over the others. While the behaviour of the gene-regulation networks alone can be determined by the Boolean dynamics [17], the overall outcome of the integrated multi-scale simulator summarising the stochastic dynamics of thousands of cells, can not be known *a priori*. By simulating hundred thousands of stochastic encountering and stimulations, we were able to perform a statistical calculation of the effective correctness of the GRN and, at the same time, to evaluate the consistency of the mesoscopic cellular rules of the agent-based model.

We executed specific simulations of the different Th differentiation routes, each with a different setup. A first *in silico* experiment was performed by stimulating the system with a generic immunogenic molecule (vaccination) at day 5 during the simulation run in the presence of lipopolysaccharide (LPS) as vaccine adjuvant.

In this system setting, the balance of Th cell sub-populations appears skewed towards IFN-γ-producing Th1 lymphocytes (fig. 2, panel C). This Th cell polarization is in well agreement with the reported ability of LPS to promote a Th1 response independently from release of inflammatory cytokines from DCs [40]. In line with the Th1 response, a phenotypic switch of macrophages (MA) towards a pro-inflammatory (MA1) as opposed to anti-inflammatory (MA2) profile [41] is observed in the dynamics of the MA population (fig. 2, panel D). Following phagocytising and presentation by specialized antigen presenting cells, the system responds by producing high-affinity immunoglobulins (fig. 2, panel A), which will mediate elimination of the antigen.

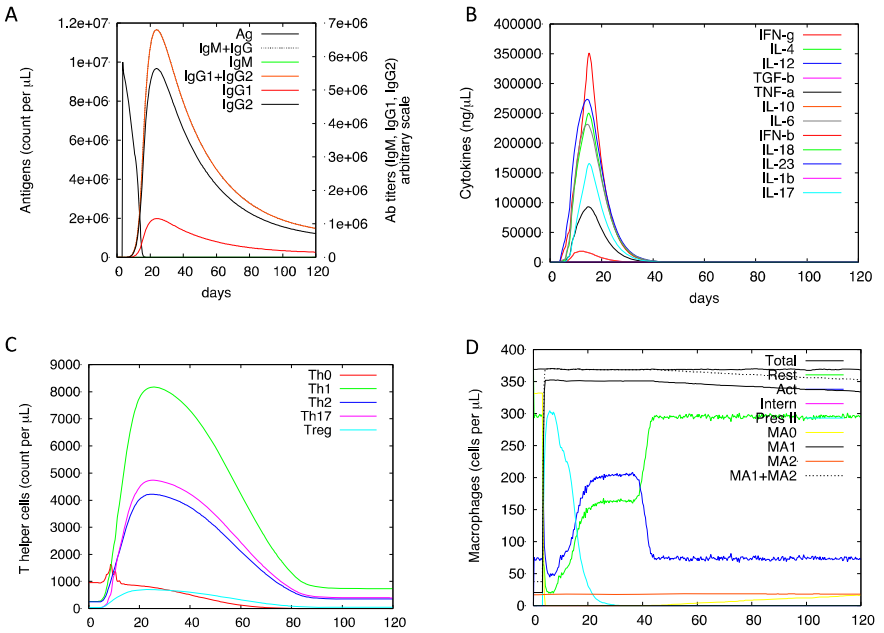


Fig. 2. Simulation of vaccination (i.e. injection of a generic immunogenic molecule) in the absence of cytokine injection. At the time of the vaccination (day 5), the steep rise in the antigen quantity is observed, followed by expanded production of antibodies, mainly of IgG type (Panel A). Increased immunoglobulin concentration shortly precedes the rapid decrease of antigen concentration until its complete clearance (day 20) likely due to effectiveness of the mounted immune response (panel A). A boost of both pro- and anti-inflammatory cytokines is observed (panel A) following antigen injection (day 5; panel B). The use of LPS as vaccine adjuvant in the simulation induces the Th1 polarisation in the CD4+ T cell population, which peaks after day 20 (panel C, green line). Presentation of the antigen on MHC-II complexes by specialized antigen presenting cells constitutes an early event in the simulated immune response to vaccination (day 5) and parallel a boost in MA1 (pro-inflammatory) macrophages which is in line with the Th1 response observed in the T helper population (panel D). Starting from day 40, contraction of the immune response appears completed and the system is back to the initial state. Accordingly, resting antigen-presenting cells now prevail on activated ones (panel D, green line).

Pathogens direct the induction of the different Th cell subtypes via distinct effects on innate immune cells. In order to test the suitability of our multiscale model we performed a series of numerical experiments aiming at inducing a different Th response. Most of the stimuli have been directly modelled in the simulation (e.g., DCs secreting IL-12, NK cells secreting IFN-γ) whereas others are indirectly taken into account (e.g., basophil production of IL-4 rendered by "injecting" a coherent quantity of IL-4 in the system). The four types of stimuli drive the system towards four different Th polarization states and transient Th subset dominance (fig. 3) [13], as follows:

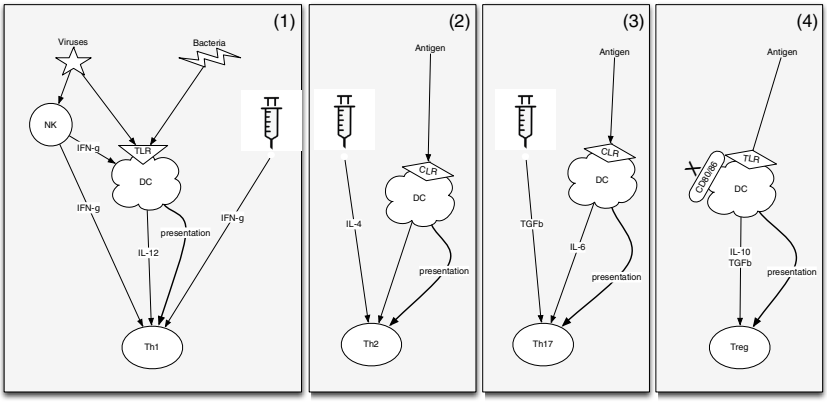


Fig. 3. Outline of four different simulation experiments performed *in silico* to test the capabilities of the multiscale model: response to viruses or intracellular bacteria eliciting Th1 polarization (panel 1), to helminth-derived products (towards Th2 phenotype, panel 2), to fungal species (towards Th17, panel 3), and to generic antigens in absence of co-stimulatory signals (towards Treg, panel 4). See main text for further details. The presence of infectious agents and associated tissue damage engage crucial pattern-recognition receptors (PRRs) at the surface of dendritic cells which recognise pathogen-associated molecular patterns (PAMPs) from the invaders, as well as danger-associated molecular patterns (DAMPs) released from damaged cells. Along with the antigen presentation by activated DCs, stimuli such as the local concentration of cytokines and DC-expressed co-stimulatory molecules will ultimately determine the development of a particular Th phenotype [13]. According such known events leading to Th differentiation [13], the different simulation experiments have been performed.

(1) Viruses or intracellular bacteria infecting target cells (e.g., epithelial cells) trigger secretion of danger signal when infected cells become necrotic. The sensing of a danger signal stimulates production of IFN-γ by NK cells. TLRs (implicitly represented in the model) expressed by dendritic cells recognise bacterial and viral PAMPs and drive IL-12 production. IFN-γ from innate immune cells such as NK cells augments the production of IL-12. IFN-γ and IL-12 cues drive the activated undifferentiated Th0 to the Th1 phenotype. In order to simulate the wave of IFN-γ raised by an intracellular microbial infection, we stimulated TLRs at the DC outer membrane by injection of IFN-γ. As expected based on *in vitro* studies, IFN-γ stimulus resulted in the in a Th1-polarized response as testified by the predominance of Th1 cells over other Th subtypes in the simulation [42] (fig. 4 panel A).

(2) To simulate helminth-derived products such as the SEA Omega-1 glycoprotein, which activates DCs to drive Th2 cell induction, we simulated CLRs engagement on DCs by injection of a binding antigen. Both IL-4-dependent and -independent induction of Th2 response have been reported, with basophils constituting the main producers of IL-4 in the former case [13]. To simulate IL-4-dependent induction of Th2 cells differentiation, injection of an amount of IL-4 which is a potent inducer of the Th2 response [43]. Accordingly, the simulation showed a prevalent expansion of the Th2 compartment following injection of IL-4 (fig. 4, panel B).

(3) Several fungal species initiate cytokine production by DCs that drive Th17 pro-inflammatory phenotype. We indirectly simulated fungal infection by providing high concentrations of TGF-β in parallel with CLR stimulation by generic antigens to reproduce a Th17 response. Signalling induced by engaging CLRs resulted in DCs secreting IL-6 and IL-23, two interleukins promoting differentiation and survival of the Th17 T cell subset, respectively [44]. Simulation of the presence of TGF-β and IL-6 in the cytokine environment (fig. 4, panel C) correctly recapitulated the induction of a Th17 response.

(4) As part of immune response dampening and homeostasis mechanisms, control processes to restrain this response from becoming detrimental are activated. Key executors of this immune suppressive function are Treg cells [12]. To obtain a protective Treg response we challenged the system with a generic antigen, which does not bear the CD80/86 ligand co-stimulatory signals for DCs, thus stimulating them to release IL-10 and TGF-β. Under these condition (i.e., the presence of TGF-β and the absence of anti-inflammatory cytokine IL-6), we observed a prevalent expansion of the Treg compartment (Figure 4, panel D) which is in agreement with experimental data [45].

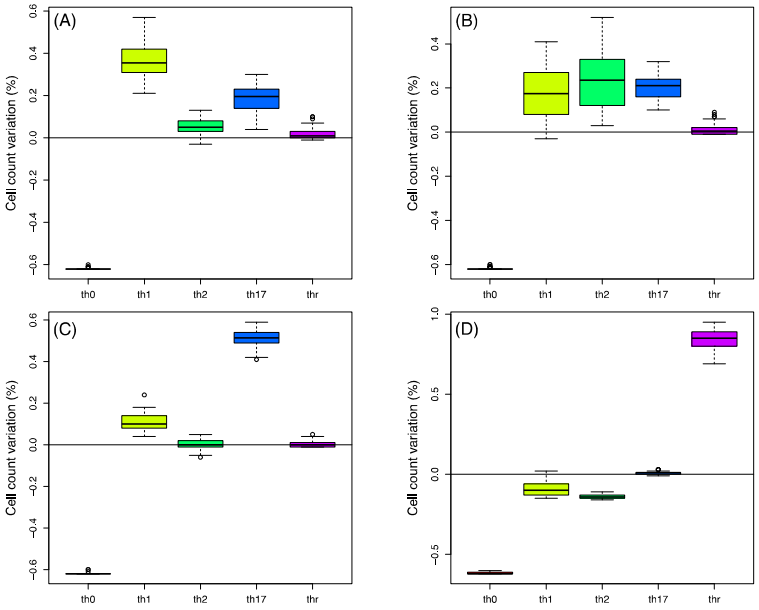


Fig. 4. Difference in Th populations following different immunogenic stimuli. The box plots A to D show the percentage difference for Th cell count from the initial baseline value at day 120 (on y-axis difference in percentage is shown: 100 * (T(t')-Th(t_0)) / T(t_0), where T stands for counts of Th1, Th2, Th17 and Treg, t_0 = 0 and T' is day 120 of the in silico experiment; statistics are computed on 50 independent runs). Panels refer to the four different experiments as in fig. 2, i.e., stimuli from viruses and intracellular bacteria trigger Th1 polarization (panel A), helmints trigger a prevalent Th2 fate (panel B), fungi trigger Th17 (panel C) and finally immune dampening by TGF-β results in Treg differentiation (panel D).

4 Conclusion

CD4+ T lymphocyte populations play key functions in the immune response to pathogens and tissue damage including help to activate CD8+ effector T lymphocytes and macrophages, support to the production of antibodies by B cells and limitation of immune responses within non-detrimental boundaries. Accordingly, a number of disorders in human and mice have been related to deregulated T helper differentiation process and cell function [2]. The founding Th1/Th2 paradigm of Th cell differentiation has been long known [9] but it is only in recent years that new fundamental Th cell subtypes have been discovered [12] [44] and the complexity and plasticity of this key immunological process have gained much appreciation.

We here implemented a multiscale computational approach to simulate T helper lymphocyte differentiation in which two different level of description, i.e., gene regulation and cell population dynamics are combined into a complex immune system model. We showed how such model integration allows bridging a gap between gene level information and cell level population by testing the model to reproduce behaviour coherent with experimental data when challenged with different immunological challenges. Further steps toward the enhancement of the integrated model will concern the removal of explicit cytokine stimulation ("cytokine injections"), mainly by implementing further cell types (e.g., basophils) in the agent-based simulator; another feature to be built in the ABM concerns the ability of recognizing specific PAMP stimuli, that will be implemented in the near future by adding PAMP receptors able to discriminate among helminths, fungi and parasites. The variety of the CD4+ T cell differentiation framework is a field of great interest and active investigation from both an immunological and a computational point of view, as testified by recent efforts and literature [8]. As new progresses in the comprehension of the complex interlink between innate and adaptive immune responses continue to be elucidated, mathematical and computational modeling approaches are likely to be prove indispensable tools to organize, and integrate different data types, spatiotemporal scales and theoretical frameworks, and to guide in vivo experimentation to accelerate the generation of new knowledge.

References

1. Pennock, N.D., White, J.T., Cross, E.W., Cheney, E.E., Tamburini, B.A., Kedl, R.M.: T cell responses: naive to memory and everything in between. Adv. Physiol. Educ. 37, 273–283 (2013)
2. Zhu, J., Yamane, H., Paul, W.E.: Differentiation of effector CD4 T cell populations (*). Annu. Rev. Immunol. 28, 445–489 (2010)
3. Hofer, T., Nathansen, H., Lohning, M., Radbruch, A., Heinrich, R.: GATA-3 transcriptional imprinting in Th2 lymphocytes: a mathematical model. Proc. Natl. Acad. Sci. U.S.A. 99, 9364–9368 (2002)
4. Yates, A., Callard, R., Stark, J.: Combining cytokine signalling with T-bet and GATA-3 regulation in Th1 and Th2 differentiation: a model for cellular decision-making. J. Theor. Biol. 231, 181–196 (2004)

5. Hong, T., Xing, J., Li, L., Tyson, J.J.: A mathematical model for the reciprocal differentiation of T helper 17 cells and induced regulatory T cells. PLoS Comput. Biol. 7, e1002122 (2011)
6. Carbo, A., Hontecillas, R., Kronsteiner, B., Viladomiu, M., Pedragosa, M., Lu, P., Philipson, C.W., Hoops, S., Marathe, M., Eubank, S., Bisset, K., Wendelsdorf, K., Jarrah, A., Mei, Y., Bassaganya-Riera, J.: Systems modeling of molecular mechanisms controlling cytokine-driven CD4+ T cell differentiation and phenotype plasticity. PLoS Comput. Biol. 9, e1003027 (2013)
7. Naldi, A., Carneiro, J., Chaouiya, C., Thieffry, D.: Diversity and plasticity of Th cell types predicted from regulatory network modelling. PLoS Comput. Biol. 6, e1000912 (2010)
8. Carbo, A., Hontecillas, R., Andrew, T., Eden, K., Mei, Y., Hoops, S., Bassaganya-Riera, J.: Computational modeling of heterogeneity and function of CD4+ T cells. Frontiers in Cell and Developmental Biology 2 (2014)
9. Mosmann, T.R., Cherwinski, H., Bond, M.W., Giedlin, M.A., Coffman, R.L.: Two types of murine helper T cell clone. I. Definition according to profiles of lymphokine activities and secreted proteins. J. Immunol. 136, 2348–2357 (1986)
10. Aggarwal, S., Ghilardi, N., Xie, M.-H., de Sauvage, F.J., Gurney, A.L.: Interleukin-23 promotes a distinct CD4 T cell activation state characterized by the production of interleukin-17. J. Biol. Chem. 278, 1910–1914 (2003)
11. Sakaguchi, S., Sakaguchi, N., Asano, M., Itoh, M., Toda, M.: Immunologic self-tolerance maintained by activated T cells expressing IL-2 receptor alpha-chains (CD25). Breakdown of a single mechanism of self-tolerance causes various autoimmune diseases. J. Immunol. 155, 1151–1164 (1995)
12. Sakaguchi, S., Wing, K., Onishi, Y., Prieto-Martin, P., Yamaguchi, T.: Regulatory T cells: how do they suppress immune responses? Int. Immunol. 21, 1105–1111 (2009)
13. Walsh, K.P., Mills, K.H.: Dendritic cells and other innate determinants of T helper cell polarisation. Trends Immunol. 34, 521–530 (2013)
14. Szabo, S.J., Kim, S.T., Costa, G.L., Zhang, X., Fathman, C.G., Glimcher, L.H.: A novel transcription factor, T-bet, directs Th1 lineage commitment. Cell 100, 655–669 (2000)
15. Maizels, R.M., Hewitson, J.P., Smith, K.A.: Susceptibility and immunity to helminth parasites. Curr. Opin. Immunol. 24, 459–466 (2012)
16. Stockinger, B., Veldhoen, M.: Differentiation and function of Th17 T cells. Curr. Opin. Immunol. 19, 281–286 (2007)
17. Mendoza, L., Pardo, F.: A robust model to describe the differentiation of T-helper cells. Theory Biosci. 129, 283–293 (2010)
18. Baldazzi, V., Castiglione, F., Bernaschi, M.: An enhanced agent based model of the immune system response. Cell Immunol. 244, 77–79 (2006)
19. Neumann, J.V.: Theory of Self-Reproducing Automata. University of Illinois Press (1966)
20. Burnet, F.M.: The Nobel Lectures in Immunology. The Nobel Prize for Physiology or Medicine, 1960. Immunologic recognition of self. Scand. J. Immunol. 33, 3–13 (1991)
21. Lederberg, J.: Genes and antibodies. Science 129, 1649–1653 (1959)
22. Brenner, S., Milstein, C.: Origin of antibody variation. Nature 211, 242–243 (1966)
23. Hayflick, L., Moorhead, P.S.: The serial cultivation of human diploid cell strains. Exp. Cell Res. 25, 585–621 (1961)
24. Schwartz, R.H.: T cell anergy. Annu. Rev. Immunol. 21, 305–334 (2003)
25. Nossal, G.J., Pike, B.L.: Clonal anergy: persistence in tolerant mice of antigen-binding B lymphocytes incapable of responding to antigen or mitogen. Proc. Natl. Acad. Sci. U.S.A. 77, 1602–1606 (1980)
26. Matzinger, P.: Tolerance, danger, and the extended family. Annu. Rev. Immunol. 12, 991–1045 (1994)

27. Matzinger, P.: The danger model: a renewed sense of self. Science 296, 301–305 (2002)
28. Jerne, N.: Towards a network theory of the immune system. Ann Immunol. 125C, 373–389 (1974)
29. Welsh, R.M., Selin, L.K.: Attrition of memory CD8 T cells. Nature 459, E3-4; discussion E4 (2009)
30. Bahl, K., Kim, S.-K., Calcagno, C., Ghersi, D., Puzone, R., Celada, F., Selin, L.K., Welsh, R.M.: IFN-induced attrition of CD8 T cells in the presence or absence of cognate antigen during the early stages of viral infections. J. Immunol. 176, 4284–4295 (2006)
31. Castiglione, F., Pappalardo, F., Bernaschi, M., Motta, S.: Optimization of HAART with genetic algorithms and agent-based models of HIV infection. Bioinformatics 23, 3350–3355 (2007)
32. Castiglione, F., Duca, K., Jarrah, A., Laubenbacher, R., Hochberg, D., Thorley-Lawson, D.: Simulating Epstein-Barr virus infection with C-ImmSim. Bioinformatics 23, 1371–1377 (2007)
33. Clancy, T., Pedicini, M., Castiglione, F., Santoni, D., Nygaard, V., Lavelle, T.J., Benson, M., Hovig, E.: Immunological network signatures of cancer progression and survival. BMC Med. Genomics 4, 28 (2011)
34. Woelke, A.L., von Eichborn, J., Murgueitio, M.S., Worth, C.L., Castiglione, F., Preissner, R.: Development of immune-specific interaction potentials and their application in the multi-agent-system VaccImm. PLoS One 6, e23257 (2011)
35. Mendoza, L.: A network model for the control of the differentiation process in Th cells. Biosystems 84, 101–114 (2006)
36. Mendoza, L., Pardo, F.: A robust model to describe the differentiation of T-helper cells. Theory Biosci. 129, 283–293 (2010)
37. Martínez-Sosa, P., Mendoza, L.: The regulatory network that controls the differentiation of T lymphocytes. Biosystems 113, 96–103 (2013)
38. Santoni, D., Pedicini, M., Castiglione, F.: Implementation of a regulatory gene network to simulate the TH1/2 differentiation in an agent-based model of hypersensitivity reactions. Bioinformatics 24, 1374–1380 (2008)
39. Janeway Ca Jr., T.P.W.M., et al.: Immunobiology: The Immune System in Health and Disease, 5th edn. Garland Science (2001)
40. Watanabe, S., Inoue, J.: Intracellular delivery of lipopolysaccharide induces effective Th1-immune responses independent of IL-12. PLoS One 8, e68671 (2013)
41. Martinez, F.O., Sica, A., Mantovani, A., Locati, M.: Macrophage activation and polarization. Front Biosci. 13, 453–461 (2008)
42. Duncan, D.D., Swain, S.L.: Role of antigen-presenting cells in the polarized development of helper T cell subsets: evidence for differential cytokine production by Th0 cells in response to antigen presentation by B cells and macrophages. Eur. J. Immunol. 24, 2506–2514 (1994)
43. Seder, R.A., Paul, W.E., Davis, M.M., de St. Groth, B.F.: The presence of interleukin 4 during in vitro priming determines the lymphokine-producing potential of CD4+ T cells from T cell receptor transgenic mice. J. Exp. Med. 176, 1091–1098 (1992)
44. Veldhoen, M., Hocking, R.J., Atkins, C.J., Locksley, R.M., Stockinger, B.: TGFbeta in the context of an inflammatory cytokine milieu supports de novo differentiation of IL-17-producing T cells. Immunity 24, 179–189 (2006)
45. Chen, W., Jin, W., Hardegen, N., Lei, K.-J., Li, L., Marinos, N., McGrady, G., Wahl, S.M.: Conversion of peripheral CD4+CD25- naive T cells to CD4+CD25+ regulatory T cells by TGF-beta induction of transcription factor Foxp3. J. Exp. Med. 198, 1875–1886 (2003)

Scaffolding of Ancient Contigs and Ancestral Reconstruction in a Phylogenetic Framework

Nina Luhmann[1], Cedric Chauve[2], Jens Stoye[1], and Roland Wittler[1]

[1] International Research Training Group "Computational Methods for the Analysis of the Diversity and Dynamics of Genomes" and Genome Informatics, Faculty of Technology and Center for Biotechnology, Bielefeld University, Germany
[2] Department of Mathematics, Simon Fraser University, Burnaby (BC), Canada

Abstract. Ancestral genome reconstruction is an important step in analyzing the evolution of genomes. Recent progress in sequencing ancient DNA led to the publication of so-called paleogenomes and allows the integration of this sequencing data in genome evolution analysis. However, the assembly of ancient genomes is fragmented because of DNA degradation over time. Integrated phylogenetic assembly addresses the issue of genome fragmentation in the ancient DNA assembly while improving the reconstruction of all ancient genomes in the phylogeny. The fragmented assembly of the ancient genome can be represented as an assembly graph, indicating contradicting ordering information of contigs.

In this setting, our approach is to compare the ancient data with extant finished genomes. We generalize a reconstruction approach minimizing the Single-Cut-or-Join rearrangement distance towards multifurcating trees and include edge lengths to avoid a sparse reconstruction in practice. When also including the additional conflicting ancient DNA data, we can still ensure consistent reconstructed genomes.

1 Introduction

In comparative genomics, one aim is to analyze the diversity of genomes from present-day species to reconstruct the structure of ancient genomes and shed light on the dynamics of evolutionary processes underlying the development of extant genomes. The speciation history leading to the present-day genomes can be represented as a phylogenetic tree. Genome reconstruction methods aim to infer genomic features, such as gene order, at internal nodes of the tree by comparing conserved features in the extant genomes at its leaves, e. g. under parsimony assumptions. This problem has already been widely studied under different models and distance formulations [1, 2, 4, 6, 9, 12, 13].

Besides the phylogeny and the genome sequences of extant species, a third source of data for reconstruction became available recently. Due to the progress in sequencing technologies, ancient DNA (aDNA) found in conserved remains can be sequenced. One example is the genome of the ancestor of *Yersinia pestis* strains that is understood to be the cause of the Black Death pandemic [3]. However, enviromental conditions influence sources for paleogenomes and result in

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 135–143, 2014.

degradation and fragmentation of DNA molecules over time, causing sequencing to produce very short reads [5]. This entails the assembly of aDNA to be specifically challenging and leads to a fragmented assembly with many short contigs requiring additional scaffolding. The purpose of the present work is to present a scaffolding method adapted to such datasets, within a phylogenetic framework.

So far, the only existing method specifically targeted at scaffolding aDNA contigs is FPSAC [11]. It follows a local approach concentrating on one internal node representing the ancestor of interest and was able to obtain a single scaffold from a fragmented assembly of the ancient *Yersinia pestis* strain. In this paper, we present a global approach for reconstructing all ancient genomes along a given phylogeny while also scaffolding the aDNA contigs obtained from a preliminary assembly for one internal node of the phylogeny. Contrary to FPSAC, our approach is global and can be described as an extension of the exact small parsimony algorithm minimizing the Single-Cut-or-Join distance described in [6] to the case of multifurcating phylogenetic trees with edge lengths. We show how this allows to handle, still with an exact polynomial time algorithm, constraints from the assembly graph of a sequenced ancestral genome.

2 Background

As a basis of this work, the data representation is described first, before the small parsimony problem under rearrangement distances is introduced.

2.1 Genome Representation

Both extant and ancient genomes are sets of chromosomes, plasmids or contigs. Each such component is represented by a sequence of oriented markers corresponding to homologous sequences, while each marker is contained once. Markers can be defined by alignment of assembled aDNA contigs onto the extant genomes (see [11] for example). To represent the orientation, any marker a has two extremities, a head a_h and a tail a_t. The order of markers in the genome can also be represented by adjacencies, which are unordered pairs of two extremities from neighboring markers, for example $\{a_h, b_t\}$. When one extremity is contained in two different adjacencies, these are said to be *conflicting*. Otherwise the genome can be written as a set of linear or circular sequences of markers and is *consistent*.

2.2 Augmented Phylogenetic Tree

The underlying general data structure for our studies, shown in Figure 1, is a phylogenetic tree $T = (V_T, E_T)$ representing the relations between extant species. Leaves annotated with assembled genome sequences correspond to extant species, internal nodes represent ancestral species. Edges are labeled with lengths describing the evolutionary distances in the tree. Furthermore, we assume that one internal node is augmented with an assembly graph $A = (V_A, E_A)$. We will refer to this augmented node as the *assembly graph node* and to the tree

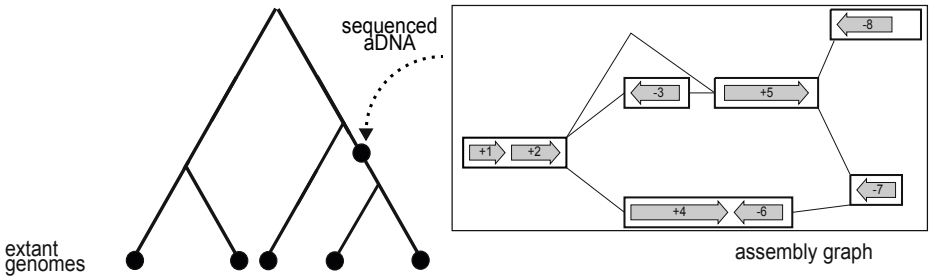


Fig. 1. Phylogenetic tree annotated with extant genomes at its leaves. One internal node is augmented with an assembly graph illustrating the fragmented assembly. It may contain conflicting adjacencies, e. g. $(2_h, 3_h)$ and $(2_h, 4_t)$.

together with the assembly graph as an *augmented phylogenetic tree*. An assembly graph is usually a de Bruijn or string graph connecting contiguous regions in the read data. Paths in the graph are then possible substrings, while branches indicate uncertainty about the exact genome sequence. For our purpose it is important to notice that since branching nodes in the graph connect one extremity with several others, they induce conflicting adjacencies.

2.3 Small Parsimony Problem under Rearrangement Distances

In order to reconstruct ancient genomes, we are starting from consistent genomes at the leaves of the considered phylogeny, represented by sets of adjacencies, and look for an optimal labeling, defined as a labeling minimizing a chosen genomic distance over the tree. This problem is known as the *small parsimony problem*.

Definition 1 (Parsimonious labeling). *Given a tree $T = (V_T, E_T)$ with each leaf l labeled with a state $b_l \in \{0, 1\}$, a labeling $\lambda : V \to \{0, 1\}$ with $\lambda(l) = b_l$ for each leaf l is* parsimonious *if it minimizes the overall distance d in the tree:*

$$W(\lambda, T) = \sum_{(u,v)\in E} d(\lambda(u), \lambda(v)).$$

In a simple setting, the distance is 0 if the label does not change along an edge and it is 1 otherwise. While for most rearrangement distances, the parsimonious labeling problem is NP-hard, one exception is the Single-Cut-or-Join (SCJ) distance introduced by Feijão and Meidanis [6], a set-theoretic rearrangement distance modeling *cuts* and *joins* of adjacencies.

Definition 2 (SCJ distance [6]). *Given two genomes defined by sets of adjacencies A and B, the SCJ distance between these genomes is*

$$d_{SCJ}(A, B) = |A - B| + |B - A|.$$

An SCJ minimizing consistent labeling over a given phylogenetic tree can be computed by the Fitch algorithm [7] in polynomial time using a binary encoding, set to 1 if the adjacency is present in the genome and set to 0 otherwise.

Reconstructed genomes then contain all adjacencies for which the internal node is labeled 1. Although adjacencies are not independent characters, it has been shown that the reconstruction of every adjacency separately assigns no conflicting adjacencies, provided that labels at all leaves are consistent and 0 is chosen in case of ambiguity at the root. The labeling for the rest of the tree is then unambiguous and provides valid genomes at internal nodes in polynomial time, minimizing the SCJ distance [6].

However, this reconstruction is sparse as it finds only the most fragmented under all co-optimal solutions. Some adjacencies will be excluded from the reconstructed genomes, although they could be included without causing conflicts. Furthermore, the Fitch algorithm can only handle binary trees and so excludes phylogenies that are not fully resolved. We will generalize the result of [6] towards multifurcating trees and show how to avoid the sparse approach. Later, in Section 4, we will show how to integrate the constraints of an assembly graph at a single ancestral node of the tree.

3 Edge-Weighted Parsimony Problem Minimizing SCJ

Like the Fitch algorithm [7], the Hartigan algorithm [8] consists of a bottom-up and a top-down traversal of the phylogenetic tree. It is a generalization towards multifurcating trees and finds, in contrast to Fitch, *all* optimal parsimonious labelings. However, this more general algorithm induces ambiguity also at internal nodes of the tree. For the small parsimony problem with the SCJ distance, it can easily be shown that choosing 0 whenever possible, also at internal nodes of the tree, results in a consistent labeling, but this could result in an even sparser solution. Conversely, always including an adjacency in case of ambiguity can result in complex conflicts and would therefore require a subsequent conflict resolving step that is mindful of the tree structure. To avoid this, we propose to include edge lengths in the reconstruction and minimize an edge-weighted SCJ distance.

Definition 3 (Edge-weighted SCJ distance labeling problem). *Given a tree $T = (V_T, E_T)$ with each leaf labeled with adjacencies and each edge $e \in E_T$ labeled with an edge length $\ell(e)$, a labeling γ of the internal nodes of T is an* edge-weighted SCJ minimizing consistent labeling *if none of the internal nodes contains a conflict and it minimizes the overall tree distance*

$$D(\gamma, T) = \sum_{(u,v)\in E} \frac{d_{SCJ}(\gamma(u), \gamma(v))}{\ell((u,v))}.$$

3.1 Hartigan Algorithm with Edge Lengths

Consider the reconstruction for one adjacency α in a tree T. A leaf is labeled according to the absence or presence of α in its extant genome. In the bottom-up phase, every node x with children $C(x)$ is annotated recursively with two

candidate sets B_1^α and B_2^α. For both states $b \in \{0,1\}$, consider all children $y \in C(x)$ with $b \in B_1^\alpha(y)$ and let $k_x^\alpha(b) = \sum_{\substack{y \in C(x),\\ b \in B_1(y)}} \frac{1}{\ell((x,y))}$ and $K = max_b\{k_x^\alpha(b)\}$. Then, if e is the edge connecting x to its parent, let

$$B_1^\alpha(x) = \{b \mid k_x^\alpha(b) = K\}, \qquad B_2^\alpha(x) = \{b \mid k_x^\alpha(b) = K - \frac{1}{\ell(e)}\}.$$

In the top-down phase, the root r is assigned with a state $b \in B_1^\alpha(r)$. When processing node x, let b be the state assigned to it in an optimal labeling. Then a child $y \in C(x)$ is labeled as follows:

$$F^\alpha(y) = \begin{cases} \{b\} & \text{if } b \in B_1^\alpha(y) \\ \{b\} \cup B_1^\alpha(y) & \text{if } b \in B_2^\alpha(y) \\ B_1^\alpha(y) & \text{otherwise} \end{cases}$$

We note that the set B_1^α is likely to be of cardinality one and the set B_2^α is likely to be empty in real data sets, where edges are annotated with non-trivial edge lengths such as rational numbers. Therefore the second case rarely occurs and there is often no choice in the other cases. Hence in most real instances, there will be a unique most parsimonious labeling for all adjacencies.

3.2 Reconstructing Consistent Genomes

Following the proof of Lemma 6.1 in [6], we can show that also the edge-weighted Hartigan algorithm assigns consistent genomes. We still assume a sparse variant of the algorithm where the label 0 is chosen during the top-down phase any time there is an ambiguity and call it *sparse edge-weighted Hartigan algorithm.*

Lemma 1. *Given two conflicting adjacencies α and β, for each node x of a tree T labeled according to the sparse edge-weighted Hartigan algorithm, if $B_1^\alpha(x) = \{1\}$, then $B_1^\beta(x) = \{0\}$ if $\nexists$ leaf l with both $B_1^\alpha(l) = \{1\}$ and $B_1^\beta(l) = \{1\}$.*

Proof. The proof is by induction on the height h of a node x in the tree, which is the maximal length from x to any descendant leaf. For $h = 0$, the node is a leaf annotated with a consistent genome, therefore the lemma holds.

When $h \geq 1$, we assume that any node with height $g < h$ and therefore all children of x satisfy the lemma. We denote the sum of edge lengths from x to all children $y \in C(x)$ annotated with $B_1^\alpha(y) = \{0\}$ with l_0^α, to children annotated with $B_1^\alpha(y) = \{1\}$ with l_1^α and edge lengths to children annotated with $B_1^\alpha(y) = \{0,1\}$ with l_{01}^α. For the two states 0 and 1, we sum the weights to the appropriate children and have $k_x^\alpha(0) = l_0^\alpha + l_{01}^\alpha$ and $k_x^\alpha(1) = l_1^\alpha + l_{01}^\alpha$. Now we can make some observations about the relation of sets B_1 in the children of x according to the Hartigan algorithm. $B_1^\alpha(x) = \{1\}$ only if $k_x^\alpha(1) > k_x^\alpha(0)$ and thus $l_1^\alpha + l_{01}^\alpha > l_0^\alpha + l_{01}^\alpha \Rightarrow l_1^\alpha > l_0^\alpha$.

Next we consider the second adjacency β that is in conflict with α. As by induction hypothesis any child y satisfies the lemma, at least all children with

$B_1^\alpha(y) = \{1\}$ have to have $B_1^\beta(y) = \{0\}$ and thus $l_0^\beta \geq l_1^\alpha$. On the other hand, only children with $B_1(\alpha, y) = \{0\}$ can have $B_1(\beta, y) = \{1\}$, so $l_1^\beta \leq l_0^\alpha$.

Taking these three observations together, we can derive that $l_0^\beta \geq l_1^\alpha > l_0^\alpha \geq l_1^\beta$, therefore $l_0^\beta + l_{01}^\beta > l_1^\beta + l_{01}^\beta$, which is the same as $k_x^\beta(0) > k_x^\beta(1)$, implying $B_1^\beta(x) = \{0\}$. Therefore when $B_1^\alpha(x) = \{1\}$, we have $B_1^\beta(x) = \{0\}$. □

Lemma 2. *Given two conflicting adjacencies α and β, for each node x of a tree T labeled according to the sparse edge-weighted Hartigan algorithm, if $F^\alpha(x) = \{1\}$, then choosing $F^\beta(x) = \{0\}$ is always possible.*

Proof. As the edge weights are only influencing the bottom-up phase, we do not have to consider them in the top-down phase. Suppose there are internal nodes with value 1 assigned to both α and β. Choose such a node with minimal distance to the root and call it v. We have different possibilities for B_1 and B_2 of v according to α and β, summarized in Table 1. Note that by Lemma 1, $B_1^\alpha(v) = B_1^\beta(v) = \{1\}$ cannot occur. In cases 1–4, choosing 0 for α or β is always possible independent of the parent assignment. In case 5, the parent assignment for both α and β has to be 1 in order to also assign 1 to v. This, however, contradicts the minimality of the depth of v and therefore concludes the proof. □

Table 1. Case differentiation for bottom-up sets that fulfill Lemma 1 and could assign label 1 for both adjacencies α and β

1	2	3	4	5
$B_1^\alpha(v) = \{1\}$	$B_1^\alpha(v) = \{0\}$	$B_1^\alpha(v) = \{0\}$	$B_1^\alpha(v) = \{1, 0\}$	$B_1^\alpha(v) = \{1, 0\}$
$B_2^\alpha(v) = \{0\} \vee \emptyset$	$B_2^\alpha(v) = \{1\}$	$B_2^\alpha(v) = \{1\}$	$B_2^\alpha(v) = \emptyset$	$B_2^\alpha(v) = \emptyset$
$B_1^\beta(v) = \{0\}$	$B_1^\beta(v) = \{1\}$	$B_1^\beta(v) = \{0\}$	$B_1^\beta(v) = \{0\}$	$B_1^\beta(v) = \{1, 0\}$
$B_2^\beta(v) = \{1\}$	$B_1^\beta(v) = \{0\} \vee \emptyset$	$B_1^\beta(v) = \{1\}$	$B_2^\beta(v) = \{1\}$	$B_2^\beta(v) = \emptyset$

Theorem 1. *For a rooted phylogenetic tree T with leaves annotated with consistent genomes containing the same set of markers, the sets $G_v = \{\alpha : F^\alpha(v) = 1\}$ assigned to all internal nodes v with the sparse edge-weighted Hartigan algorithm are consistent genomes and minimize the edge-weighted SCJ distance.*

Proof. According to Theorem 6.3 in [6], including the adjacency α in every node v, where $F^\alpha(v) = 1$, builds genomes that minimize the SCJ distance over T. Lemma 2 shows that also with the sparse edge-weighted Hartigan algorithm no conflicting adjacencies will be assigned to a node v. Therefore the sets G_v minimize the total sum of SCJ cost per edge length. □

4 Integrating aDNA Sequencing Information

The assembly graph based on ancient sequencing reads (cf. Figure 1) defines putative adjacencies between markers on connected contigs. These adjacencies constrain the reconstruction by providing evidence of the genome structure directly

seen at an internal point in the tree. We include these constraints by extending the original tree with an additional leaf attached to the assembly graph node. This leaf will be labeled with the presence or absence of an adjacency in the assembly graph, just like the leaves representing extant genomes. The respective edge length $\ell(e)$ has to be chosen in regard to the other two connected edges a and b of the assembly graph node such that the additional information is relevant at all but not generally dominating. Hence it has to be chosen such that $\frac{1}{\ell(e)}$ is in the interval $[(\frac{1}{\ell(a)} - \frac{1}{\ell(b)}), (\frac{1}{\ell(a)} + \frac{1}{\ell(b)})]$ for $\ell(a) < \ell(b)$, where a smaller edge length gives the assembly graph more importance.

However the set of adjacencies present in the assembly graph is not necessarily consistent and can cause conflicts. Instead of adding a postprocessing step that resolves all the conflicts in the tree after the reconstruction, we propose in Algorithm 1 an approach that integrates the conflicts resolving into the reconstruction process. To resolve conflicts, we rely on the exact polynomial time MAX-ROW-component-mCi1P algorithm described in [10]. It selects a subset of adjacencies that form a set of linear and/or circular chromosomes based on a maximum-weight matching in a graph.

Algorithm 1. Consistent reconstruction integrating aDNA sequencing data

Input: A tree T with edge lengths, extant consistent genomes, aDNA assembly graph
Output: A consistent labeled tree minimizing the edge-weighted SCJ distance.

```
Attach an additional leaf to the assembly graph node v
Reroot the tree such that v becomes its root
for each adjacency α do
    for each internal node x in T do
        Compute B_1^α(x) and B_2^α(x) with the sparse edge-weighted Hartigan algorithm
A = {α|1 ∈ B_1^α(v)}
Solve MAX-ROW-component-mCi1P for A
for each adjacency α do
    for each internal node x in T do
        Compute F^α(x) with the sparse edge-weighted Hartigan algorithm
```

Theorem 2. *Given an augmented phylogenetic tree, Algorithm 1 computes a consistent labeling integrating the assembly graph information and minimizing the edge-weighted SCJ distance in polynomial time.*

Proof. Acoording to Theorem 1, the sparse edge-weighted Hartigan algorithm assigns consistent, SCJ minimizing genomes when the leaf labels are consistent. Rerooting the tree will not affect the outcome of the reconstruction. In the bottom-up phase, the conflicting leaf will only influence the assignment at the root. All other internal nodes fulfill Lemma 1, as the original leaves are consistently labeled. Therefore they cannot cause a conflicting assignment in the top-down phase when the parent assignment is consistent. As conflicts can thus only occur at the root node, they have to be resolved with a minimal increase in parsimony costs before propagating the assignment down the tree during the top-down phase. Selecting a maximum cardinality subset of all adjacencies assigned

to the root can be done by solving the MAX-ROW-component-mCi1P [10]. With a consistent root labeling, the top-down assignment will be consistent according to Lemma 2.

The traversal of the tree with n leaves and a adjacencies takes $O(an)$ time. The MAX-ROW-component-mCi1P can be solved in $O(a^{3/2})$ [10]. Therefore the overall running time is in $O(an + a^{3/2})$. □

5 Conclusion

We have described a generalization of the exact algorithm solving the small parsimony problem under the SCJ rearrangement distance. Computing the labeling of internal nodes with the Hartigan algorithm enables handling multifurcating trees. Including edge lengths still ensures the reconstruction of valid genomes and is also expected to provide a unique optimal solution under non-trivial edge lengths in practice. Building upon this result, we presented an integrated phylogenetic assembly approach. It includes aDNA sequencing information in the reconstruction of other ancient genomes in the phylogeny and also scaffolds the fragmented assembly while minimizing the SCJ distance.

Among the questions our work raises, it would be interesting to study model variants that allow to integrate copy numbers or unequal marker content. Another question of interest would be to design efficient heuristics or parameterized algorithms to augment an initial parsimonious consistent labeling with extra adjacencies that preserve both parsimony and consistency.

Acknowledgements. NL and RW are funded by the International DFG Research Training Group GRK 1906/1.

References

[1] Bergeron, A., Blanchette, M., Chateau, A., Chauve, C.: Reconstructing ancestral gene orders using conserved intervals. In: Jonassen, I., Kim, J. (eds.) WABI 2004. LNCS (LNBI), vol. 3240, pp. 14–25. Springer, Heidelberg (2004)

[2] Bertrand, D., Gagnon, Y., Blanchette, M., El-Mabrouk, N.: Reconstruction of ancestral genome subject to whole genome duplication, speciation, rearrangement and loss. In: Moulton, V., Singh, M. (eds.) WABI 2010. LNCS (LNBI), vol. 6293, pp. 78–89. Springer, Heidelberg (2010)

[3] Bos, K.I., Schuenemann, V.J., Golding, G.B., et al.: A draft genome of yersinia pestis from victims of the black death. Nature 478(7370), 506–510 (2011)

[4] Chauve, C., Tannier, E.: A methodological framework for the reconstruction of contiguous regions of ancestral genomes and its application to mammalian genomes. PLoS Computational Biology 4(11), e1000234 (2008)

[5] Drancourt, M., Raoult, D.: Palaemicrobiology: current issues and perspectives. Nature Rev. Microbiol. 3, 23–35 (2005)

[6] Feijão, P., Meidanis, J.: SCJ: a breakpoint-like distance that simplifies several rearrangement problems. IEEE/ACM Trans. Comput. Biol. and Bioinf. 8(5), 1318–1329 (2011)

[7] Fitch, W.M.: Toward defining the course of evolution: minimum change for a specific tree topology. Systematic Biology 20(4), 406–416 (1971)
[8] Hartigan, J.A.: Minimum mutation fits to a given tree. Biometrics, 53–65 (1973)
[9] Ma, J., Zhang, L., Suh, B.B., et al.: Reconstructing contiguous regions of an ancestral genome. Genome Res. 16(12), 1557–1565 (2006)
[10] Maňuch, J., Patterson, M., Wittler, R., Chauve, C., Tannier, E.: Linearization of ancestral multichromosomal genomes. BMC Bioinformatics 13(19), S11 (2012)
[11] Rajaraman, A., Tannier, E., Chauve, C.: FPSAC: fast phylogenetic scaffolding of ancient contigs. Bioinformatics 29(23), 2987–2994 (2013)
[12] Stoye, J., Wittler, R.: A unified approach for reconstructing ancient gene clusters. IEEE/ACM Trans. Comput. Biol. Bioinf. 6(3), 387–400 (2009)
[13] Zheng, C., Sankoff, D.: On the pathgroups approach to rapid small phylogeny. BMC Bioinformatics 12(S-1), S4 (2011)

Quality Metrics for Benchmarking Sequences Comparison Tools

Erwan Drezen and Dominique Lavenier

INRIA/IRISA/Genscale, Campus de Beaulieu, Rennes, France
erwan.drezen@inria.fr,
dominique.lavenier@irisa.fr

Abstract. Comparing sequences is a daily task in bioinformatics and many software try to fulfill this need by proposing fast execution times and accurate results. Introducing a new software in this field requires to compare it to recognized tools with the help of well defined metrics. A set of quality metrics is proposed that enables a systematic approach for comparing alignment tools. These metrics have been implemented in a dedicated software, allowing to produce textual and graphical benchmark artifacts.

Keywords: sequence, alignment, benchmark.

1 Introduction

In Bioinformatics, the task of comparing genomic sequences is ubiquitous, leading to various software proposals over the years. With the NGS revolution, this task is becoming more and more time consuming as the size of NGS data truly explodes.

Software have tried to cope with this situation in different ways. First, by using known algorithms with optimized hardware usage. Second, by proposing new heuristics to reduce the searching space. The historically sequence comparison tools are:

- SSEARCH [1] that proposes an implementation of the Smith-Waterman algorithm with an efficient usage of multicore / SSE architecture; it provides an "exact" search alignment algorithm and is therefore slow, and consequently inadapted for large requests.
- BLAST [2] and FASTA [1] that both propose a seed based heuristics to speed-up the computation with a small loss of quality w.r.t. exact algorithms.

The trade-off between speed and quality is the key point. As a consequence, to evaluate new Alignment Search Tools (AST for short), benchmarking has to focus on time and quality metrics. Comparing execution times is straightforward. On the other hand, comparing quality is more challenging since it requires to define a precise quality metric.

S. Campos (Ed.): BSB 2014, LNBI 8826, pp. 144–153, 2014.

The paper presents a methodology for comparing the results of AST. We explicitly target intensive sequence comparison for which there is an increasing demand due to high throughput sequencing opportunities. A quality metric is defined and applied to a set of software and benchmarks.

2 Benchmarking AST

2.1 Configuration

To fairly benchmark AST, many configuration aspects must be taken into consideration:

1. For a given request, AST have to be configured for providing similar results. This is far from obvious since they generally propose a large set of parameters. In some cases, they can have hidden configuration. For instance, some AST may keep only the best N hits for a given query, some results being thus filtered out; other AST that don't use such filtering should find more alignments, making AST comparison difficult.
2. Some AST implicitly modified the input banks. For instance, low complexity region [3], [4] can be systematically masked. These transformations may have significant influence on the results compared to other tools that are not applying these kinds of pre-processing.
3. AST often provide a statistical model telling whether an alignment has to be kept or not (for instance Karlin/Altschul model of Blast). Thus, AST with different models are likely to produce different alignments.

These points have a strong impact on the alignment list generated by the AST. Consequently, they need to be included in the quality metric itself. As a side effect, the AST command lines have to be explicitly described as part of the benchmark results.

In the following, we note $T^{(0)}$ (or T for short) an AST with its default configuration. Specific configurations of T will be noted as $T^{(i)}$ for $i \geq 1$.

2.2 Input Request

To compare AST, a minimal set of common parameters needs to be specified. Table 1 lists the parameters we selected.

Parameters r_1 to r_9 have a direct impact on the output alignments. The r_{10} parameter has an impact on the execution time. The execution time can be used to analyze the scalability of AST as a function of the number of cores.

A request R can now be defined by a set of r_i. If not all r_i are specified, it is supposed that default values are used instead. A relation of equivalence can also be defined between two requests R_a and R_b:

$$R_a \sim R_b \iff \forall i \in [1..10];\ r_i(R_a) = r_i(R_b)$$

Table 1. Request parameters

r_1) query bank	r_6) reward cost
r_2) subject bank	r_7) penalty cost
r_3) e-value	r_8) substitution matrix
r_4) open gap cost	r_9) low complexity filtering
r_5) extend gap cost	r_{10}) number of cores

2.3 Output Data

Comparing AST qualities implies that AST generates the same nature of information. We choose the information provided by the Blast tabular output format; for each alignment, it provides 12 properties p_i as shown in table 2. Table 3 is an illustration of a Blast output.

Table 2. Alignment properties

p_1) query sequence id	p_7) start offset in query
p_2) subject sequence id	p_8) end offset in query
p_3) percentage of identity	p_9) start offset in subject
p_4) alignment length	p_{10}) end offset in subject
p_5) number of mismatches	p_{11}) e-value
p_6) number of gap openings	p_{12}) bitscore

Relying only on this set of properties means that only AST able to provide this information would be candidate. Fortunately, many AST can be parameterized to generate Blast-like tabular output format.

Table 3. Example of Blast tabular output

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}	p_{12}
SPO13535	SPP02309	39.34	305	181	4	29	332	17	302	7.5e-62	231.6
SPO13535	SPQ91G55	34.93	209	134	2	42	250	46	252	1.5e-31	130.4
SPP02400	SPQ196T6	30.18	603	388	3	90	663	49	647	1.6e-76	282.4

3 Alignment Quality Metric

3.1 Definitions

Before specifying an alignment quality metric, we introduce the following definitions:

- I is a set of integers $i \in [1..12]$. I specifies a subset of properties p_i defined in table 2

- For a given set I, an alignment a is an object having properties p_i for $i \in I$
- Two alignments x and y are equivalent if their properties have the same values:

$$x \sim y \iff \forall i \in I;\ p_i(x) = p_i(y)$$

- an alignment set A contains alignments defined for a specific set I

An alignment set may contain equivalent items. In such a case, items equivalent to some alignment a are grouped into an equivalence class $\tilde{a}$.

$$\tilde{a} = \{x \in A; a \sim x\}$$

We define $\tilde{A}$ as the set built from A by keeping only one item of each equivalence class $\tilde{a}$.

Taking the example of table 3, for $I = \{1\}$, we have the alignment set $A = \{a1, a2, a3\}$ and the set of equivalent alignments $\tilde{A} = \{\tilde{a_1}, \tilde{a_2}\}$ with:

$$\begin{cases} p_1(a1) = "SPO13535" \\ p_1(a2) = "SPO13535" \\ p_1(a3) = "SPP02400" \end{cases} \qquad \begin{cases} p_1(\tilde{a_1}) = "SPO13535" \\ p_1(\tilde{a_2}) = "SPP02400" \end{cases}$$

3.2 From AST Output to Alignment Set

With these definitions, an AST output can be formally transformed into an alignment set A where only some properties defined by the set I are kept. An alignment set can thus be defined as $A := \varphi(T, R, I)$ with T corresponding to the AST, R the request and I the set of properties.

Figure 1 illustrates the transformation of a Blast output.

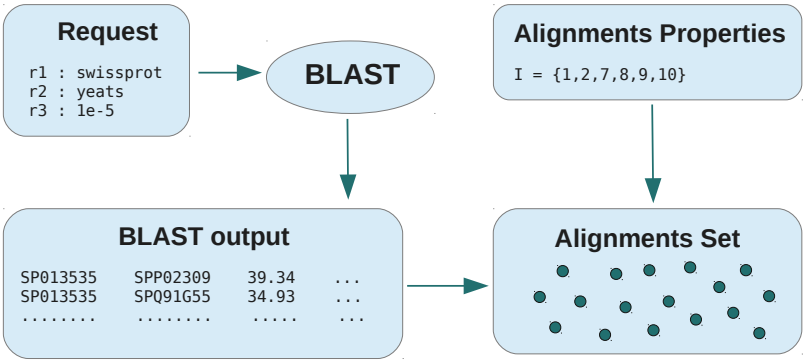


Fig. 1. Building an alignment set from an AST output

Note also that the same output generated by a tool will be viewed differently for two distinct sets I_1 and I_2. Therefore several quality metrics based on different I sets can be defined to provide different benchmark points of view.

3.3 Comparing AST

For a request R (defined by a set of r_i from table 1) and a given set I, we note:

$$A_1 = \varphi(T_1, R, I),\ alignment\ set\ found\ by\ AST\ T_1\ for\ request\ R$$
$$A_2 = \varphi(T_2, R, I),\ alignment\ set\ found\ by\ AST\ T_2\ for\ request\ R$$

Comparing T_1 to T_2 consists in finding a set $M^{(I)}$ such as:

$$M^{(I)} = \begin{cases} (x,y) \in A_1 \times A_2; & x \sim y \\ \forall (x',y') \in A_1 \times A_2; & \begin{cases} x \sim y' \Rightarrow y' = y \\ x' \sim y \Rightarrow x' = x \end{cases} \end{cases}$$

The intuitive idea of this definition is to map an item from one set to at most one item of the other set. Note that some alignments in one set may not be connected to an alignment in the other set as shown figure 2.

The estimation of "how close two sets A_1 and A_2 are" can be done by considering the cardinality $|M^{(I)}|$ which represents the number of common alignments between A_1 and A_2. We define three numbers $N_c^{(I)}$, $N_1^{(I)}$ and $N_2^{(I)}$:

- $N_c^{(I)} := |M^{(I)}|$, number of common alignments between A_1 and A_2
- $N_1^{(I)} := |A_1| - |M^{(I)}|$, number of alignments specific to A_1
- $N_2^{(I)} := |A_2| - |M^{(I)}|$, number of alignments specific to A_2

Finally, the alignment quality metric for a given I under a $Q^{(I)}$ function is defined as:

$$\boxed{A_1 \times A_2 \xrightarrow{Q^{(I)}} \left(N_c^{(I)}, N_1^{(I)}, N_2^{(I)}\right)}$$

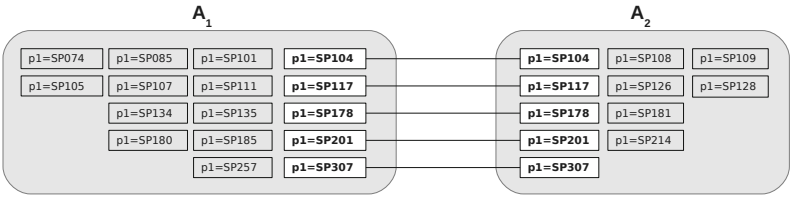


Fig. 2. Example of equivalent alignments between two sets A_1 and A_2 for $I = \{1\}$

With the following properties:

1. $Q^{(I)}(A, A) = (|A|, 0, 0)$
2. $Q^{(I)}(A_1, A_2) = (x, y, z) \Longrightarrow Q^{(I)}(A_2, A_1) = (x, z, y)$

Note that this definition of the alignment quality metric is relative: it only gives information for a specific couple of alignment sets. It does not provide absolute assessment for one AST. The consequence is that one of the AST (T_1, T_2) needs to be chosen as a reference when comparing many AST together.

The percentage of alignments found by T_2 in the alignments of T_1 can also be defined as:

$$\alpha_2^{(I)} = \frac{N_c^{(I)}}{N_c^{(I)} + N_1^{(I)}}$$

Taking figure 2 as example, we have $Q^{(I)}(A_1, A_2) = \big(5, 11, 6\big)$. T_2 is able to find $\alpha_2^{(I)} = \frac{5}{5+11} = 31.25\%$ of the alignments found by T_1. Similarly, T_1 finds $\alpha_1^{(I)} = \frac{5}{5+6} = 45.45\%$ of the alignments found by T_2.

Comparing N AST T_i is simply done by computing $Q^{(I)}(A_i, A_j)$ for all $(i, j) \in [1..N]^2$. Note that computation is only required for $i >= j$ because of the second property of the quality definition.

4 Results

4.1 List of Benchmarked Tools and Quality Metrics

The evaluation of our methodology has been done with the following tools :

T_1:=SSEARCH T_2:=BLAST T_3:=PLAST T_4:=UBLAST

In this set of AST, SSEARCH is the only exact tool, the other ones are based on heuristics. Thus, SSEARCH is taken as the reference. In that way, it can be seen how close heuristics based tools are from an exact reference. BLAST [2], PLAST [5] and UBLAST [6] are heuristic based tools.

By default, heuristics based tools are configured to get the best trade-offs between speed and quality. Of course, these tools can also be configured by setting specific parameters in order to get a better quality or a better speed. Table 4 gives several configurations used for the benchmark.

Three different quality metrics (defined by a specific set I) are considered :

- $I_{query} = \{1\}$ with $\tilde{A}$, the equivalent class alignment set; this metric compares the number of matched queries between tools
- $I_{hit} = \{1, 2\}$ with $\tilde{A}$, the equivalent class alignment set; this metric compares the number of matched couples [subject,query] between tools
- $I_{align} = \{1, 2, 7, 8, 9, 10\}$ with A, the alignment set; this metric compares the number of alignments sharing the same [subject,query] identifiers and the same [subject,query] boundaries.

Table 4. Tools specific configurations

Name	Configuration	Purpose
BLASTN	blast -task blastn	nucl/nucl requests
MEGABLAST	blast -task megablast	nucl/nucl requests
BLASTQ	blast -max-target-seqs 25000	quality improvement
UBLASTQ	ublast -accel 1	quality improvement
PLASTS	plast -seeds-use-ratio 0.01	speed improvement

These 3 metrics go from the worst to the best precision. As a matter of fact, the corresponding sets I provide more and more selective relations of equivalence between alignments. As explained later, two AST may be close for I_{query} and even for I_{hit} but may have significant differences for I_{align}; this is shown by small value for $N_c^{(I_{align})}$ and big values for $N_1^{(I_{align})}$ and $N_2^{(I_{align})}$.

4.2 Results

In the following tables, we dump $\alpha_2^{(I)}$, the percentage of alignments found by T_2 in the alignment set of T_1.

Two ratios of execution time between T_1 and T_2 can be defined:

1. SU_{total} : speedup for the total execution time of a request, including bank preparation when needed (use of *makeblastdb* for instance)
2. SU_{AST} : speedup for the execution time of the sequences comparison only (exclude *makeblastdb* execution time for instance)

Table 5. [Escherichia Coli vs. uniprot_sprot, evalue=1e-3, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
ssearch	*blast*	55.5	58.1	99.8	5.48	5.80
ssearch	*plast*	93.9	97.1	99.8	26.22	32.54
ssearch	*ublast*	29.0	64.4	99.8	64.41	146.60
ssearch	*blastQ*	94.2	97.4	99.8	4.90	5.08
ssearch	*plastS*	73.2	77.1	99.4	52.07	84.43
ssearch	*ublastQ*	30.4	69.1	99.8	40.61	62.63

Table 6. [Chitinophaga Pinensis vs. uniprot_sprot, evalue=1e-3, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
ssearch	*blast*	61.4	64.2	98.7	5.33	5.45
ssearch	*plast*	92.0	95.9	97.9	29.23	33.26
ssearch	*ublast*	19.9	52.4	78.9	89.48	157.96

Tables 5 and 6 show a protein/protein comparison between swissprot [10] and two bacterias [8],[9] from which several comments can be done:

- Heuristics based tools have good speedups, especially UBLAST
- AST quality are very close to SSEARCH for $I^{(query)}$ (except UBLAST, Table 6)
- With default configuration, BLAST and UBLAST have poor results for $I^{(hit)}$ and $I^{(align)}$; only PLAST manages to recover most of the SSEARCH results
- For two similar requests, PLAST has the smallest discrenpencies for the 3 metrics, followed by BLAST and then by UBLAST

Table 7. [uniprot_sprot_40000b vs. uniprot_sprot_40000, evalue=1e-30, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
ssearch	*blast*	86.5	89.9	97.1	8.40	8.46
ssearch	*plast*	88.9	92.4	96.3	44.62	46.48
ssearch	*ublast*	6.4	19.9	22.8	375.46	591.03

Tables 7 compares a subset of 40000 sequences from swissprot to another subset of 40000 sequences from swissprot. Here, BLAST and PLAST have similar values for the 3 metrics and are close to the reference; speedups are interesting, especially for PLAST. For this request, UBLAST is still very fast but has poor values for the 3 metrics. Other tools configurations give similar quality results.

Table 8. [SRR142736 vs. uniprot_sprot, evalue=1e-3, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
blastQ	*blast*	73.8	72.3	100.0	1.00	1.00
blastQ	*plast*	77.7	95.4	98.1	6.70	7.10
blastQ	*ublast*	19.8	64.9	82.0	40.30	68.91

Table 9. [TARA sample vs. uniprot_sprot, evalue=1e-3, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
blastQ	*blast*	82.1	80.8	100.0	1.00	1.00
blastQ	*plast*	80.8	94.7	83.1	6.50	6.68
blastQ	*ublast*	17.2	59.2	29.9	71.10	105.50

Tables 8 shows a genomic query and table 9 a meta-genomic query [11], with swissprot as subjet bank. SSEARCH can't be used as reference here so BLASTQ is used instead. BLAST and PLAST are similar for $I^{(align)}$, with better results for PLAST on $I^{(hit)}$ and better results for BLAST on $I^{(query)}$. UBLAST is still the fastest tool with poor quality, in particular in the meta-genomic case.

Table 10. [SRR027344 vs. nt 2000000 seqs, evalue=1e-30, nbcores=16]

T_1	T_2	$\alpha_2(I_{align})$	$\alpha_2(I_{hit})$	$\alpha_2(I_{query})$	SU_{total}	SU_{AST}
blastQ	*blast*	64.8	64.6	100.0	1.03	1.03
blastQ	*megablast*	61.1	61.0	96.6	5.17	12.48
blastQ	*plast*	94.2	94.2	96.8	5.38	14.07
blastQ	*ublast*	90.1	94.9	95.6	0.63	0.65

Tables 10 is a nucleotide/nucleotide request, comparing a set of reads to $2,000,000$ sequences from the nt [12] database. Here, PLAST and UBLAST provide the best quality metrics w.r.t. BLASTQ, with good speedups for PLAST.

In brief, we summarize this benchmark with the following observations:

- As expected, heuristics based tools are fast at the expense of quality; for instance, UBLAST is very fast but has often poor quality for the chosen metrics
- Surprisingly, BLAST with default configuration has sometimes poor quality for some metrics; a special configuration like BLASTQ is needed for recovering full quality
- PLAST is a good trade-off between speed and quality; it is faster than BLAST with similar quality values.

5 Conclusion

In this paper, we have defined a set of quality metrics for comparing alignment search tools (AST). Comparing AST is important to understand how the AST behave and what should be expected from the alignments they produce. In particular, these metrics allow the distance between exact and heuristics based tool to be precisely evaluated.

These metrics have been tested on standard AST. The benchmark results provides some interesting hints that can be used to match a specific tool with user needs; for instance, PLAST is a good trade-off in terms of speed/quality, and UBLAST is interesting if speed is crucial and if quality loss is acceptable (particularly for alignment and hit metrics).

Other quality metrics are currently under investigation to better capture user needs according to the application domains where AST are involved. A next step is also to gather in a public database many experimentations and offer to the scientific community a way to select the best AST according to their needs. Other AST will also be added to extend the scope of this methodology. Finally, a web interface [7] is under development and already provides a graphical representation of the metrics.

References

1. Pearson, Lipman: Improved tools for biological sequence comparison. Proc. Natl. Acad. Sci. USA, 2444–2448 (1988)
2. Altschul, Gish, Miller, Myers, Lipman: Basic local alignment search tool. J. Mol. Biol. 215, 403–410 (1990)
3. Wootton, J.C., Federhen, S.: Statistics of local complexity in amino acid sequences and sequence databases. Computers and Chemistry 17, 149–163 (1993)
4. Morgulis, Gertz, Schaffer, Agarwala: Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences
5. Nguyen, V.H., Lavenier, D.: PLAST: parallel local alignment search tool for database comparison. BMC Bioinformatics 10 (2009)
6. USearch, Ultra-fast sequence analysis, `http://drive5.com/usearch`
7. Web Interface for Metrics display, `http://irisa.lavenier.net/cast/ISCB_LA_2014`
8. Proteome of Escherichia Coli, `ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Escherichia_coli_K_12_substr__DH10B_uid20079/CP000948.faa`
9. Proteome of Chitinophaga Pinensis, `ftp://ftp.ncbi.nlm.nih.gov/genbank/genomes/Bacteria/Chitinophaga_pinensis_DSM_2588_uid27951/CP001699.faa`
10. Swissprot bank, `ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/complete/uniprot_sprot.fasta.gz`
11. Environmental samples from TARA, `ftp://ftp-trace.ncbi.nlm.nih.gov/sra/sra-instant/reads/ByRun/sra/ERR/ERR550/ERR550538/ERR550538.sra`
12. nt database, `ftp://ftp.ncbi.nlm.nih.gov/blast/db/FASTA/nt.gz`

Author Index