Marco Tomassini
Alberto Antonioni
Fabio Daolio
Pierre Buesser (Eds.)

LNCS 7824

Adaptive and Natural Computing Algorithms

11th International Conference, ICANNGA 2013
Lausanne, Switzerland, April 2013
Proceedings

Lecture Notes in Computer Science 7824

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Marco Tomassini Alberto Antonioni
Fabio Daolio Pierre Buesser (Eds.)

Adaptive and Natural Computing Algorithms

11th International Conference, ICANNGA 2013
Lausanne, Switzerland, April 4-6, 2013
Proceedings

Springer

Volume Editors

Marco Tomassini
Alberto Antonioni
Fabio Daolio
Pierre Buesser

Université de Lausanne, Faculté des Hautes Etudes Commerciales
Département des Systèmes d'Information
UNIL-Dorigny, Bâtiment Internef, 1015 Lausanne, Switzerland
E-mail: {marco.tomassini, alberto.antonioni, fabio.daolio, pierre.buesser}@unil.ch

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-37212-4 e-ISBN 978-3-642-37213-1
DOI 10.1007/978-3-642-37213-1
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2013933232

CR Subject Classification (1998): F.1-2, I.5, I.2

LNCS Sublibrary: SL 1 – Theoretical Computer Science and General Issues

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

We are pleased to present in this LNCS volume the proceedings of the 11^{th} International Conference on Adaptive and Neural Computing Algorithms, ICANNGA 2013, that was held in Lausanne, Switzerland. The biennial ICANNGA series of conferences was started in 1993 in Innsbruck, Austria, and was followed by Ales, France (1995), Norwick, UK (1997), Portorož, Slovenia (1999), Prague, Czech Republic (2001), Rouen, France (2003), Coimbra, Portugal (2005), Warsaw, Poland (2007), Kuopio, Finland (2009), and Ljubljana, Slovenia (2011). The present edition thus marks the 20^{th} year of existence of this successful series of conferences.

We received 91 paper submissions for this edition coming from many different countries. Following an extensive review process, the Program Committee selected 51 manuscripts for inclusion in this volume. Of the 51 papers, 39 were presented in oral sessions and the rest as posters. The selected papers cover many aspects of soft computing techniques and adaptive algorithms, from artificial neural networks to evolutionary algorithms, system dynamics and identification, pattern recognition, machine learning techniques, and swarm computing among others. Both theoretical and fundamental contributions as well as applications were present, although theoretical and numerical models were the majority.

The conference featured three distinguished keynote speakers: Tom Heskes, Moshe Sipper, and Alessandro Villa. Their presentations were at the leading edge of today's research and of great inspirational value. Tom Heskes' talk was about Bayesian machine learning approaches to analyze complex data sets coming from the brain's functional data and their application to brain–computer interfaces, a really exciting perspective. Moshe Sipper focused on artificial intelligence techniques based on evolutionary computation within the domain of games, an activity in which Sipper's group has produced human-competitive and award-winning game strategies in games such as chess, checkers, and several others. Alessandro Villa's talk was on theoretical models of spatio-temporal patterns of recurrent neural networks based on dynamical systems theory. He showed that only selected meaningful patterns may contribute to extend the computational power of neural networks.

The success of a conference depends on the quality of the scientific contributions, as well as on the work of the reviewers and of the organizers. We are grateful to all the contributors for their hard and high-quality work that made the conference possible. And of course, we thank the reviewers for their time and careful work. We would also like to express our gratitude to the Advisory Committee, which guarantees the continuity of the conference series and provided advice, feedback, and discussion throughout. Finally, we thank the Economics Faculty of the University of Lausanne for the logistic support provided, which

proved very important in creating a nice and productive environment for all the conference activities.

April 2013

Marco Tomassini
Alberto Antonioni
Fabio Daolio
Pierre Buesser

Organization

Advisory Committee

Rudolf Albrecht
Bartlomiej Beliczynski
Andrej Dobnikar
Mikko Kolehmainen
Vera Kurkova
David Pearson
Bernardete Ribeiro
Nigel Steele

Program Committee

Hernan Aguirre
Jarmo Alander
Rudolf Albrecht
Alberto Antonioni
Rubén Armañanzas
Wolfgang Banzhaf
Miguel Arturo Barreto-Sanz
Fülöp Bazsó
Bartlomiej Beliczynski
Lubica Benuskova
Pascal Bouvry
Hans Albert Braun
Stefano Cagnoni
Paolo Cazzaniga
Francesco Cerutti
Francisco Chicano
Carlos Coello Coello
Ernesto Costa
Carlos Cotta
Fabio Daolio
Christian Darabos
Ivanoe De Falco
Matteo De Felice
Antonio Della Cioppa
Federico Divina
Andrej Dobnikar
Bernabe Dorronsoro
António Dourado
Francisco Fernández de Vega
Stefan Figedy
Alexandru Floares
Mario Giacobini
Michele Giugliano
Juan A. Gomez-Pulido
Barbara Hammer
Jin-Kao Hao
Ignacio Hidalgo
Osamu Hoshino
Lazaros Iliadis
Marcin Iwanowski
Juan Luis Jimenez Laredo
Martti Juhola
Paul Kainen
Mario Koeppen
Mikko Kolehmainen
Stefanos Kollias
Petia Koprinkova-Hristova
Jozef Korbicz
Vera Kurkova
Giancarlo La Camera
Kauko Leiviskä
Tom Lenaerts
Uros Lotric
Francesco Masulli
Julian Miller
Francesco C. Morabito
Alberto Moraglio
Juan Manuel Moreno
Ferrante Neri
Roman Neruda

Ernst Niebur
Stanislaw Osowski
Jorge Peña
Carlos Andrés Peña-Reyes
Andrés Pérez-Uribe
Clara Pizzuti
Riccardo Poli
Mike Preuss
Bernardete Ribeiro
Conor Ryan
Jorge Santos
Henrik Saxen
Marc Schoenauer
Roberto Serra
Catarina Silva
Moshe Sipper
Branko Ster
Thomas Stuetzle
Miroslaw Swiercz
Ryszard Tadeusiewicz
El-Ghazali Talbi
Tatiana Tambouratzis
Marco Tomassini
Leonardo Vanneschi
Miguel A. Vega-Rodríguez
Sébastien Verel
Alessandro Villa
Stefan Wermter

Organizing Committee

Alberto Antonioni
Pierre Buesser
Fabio Daolio
Elisabeth Fournier Pulfer
Marco Tomassini

Table of Contents

On Appropriate Refractoriness and Weight Increment in Incremental Learning

Toshinori Deguchi[1], Junya Fukuta[1], and Naohiro Ishii[2]

[1] Gifu National College of Technology
[2] Aichi Institute of Technology

Abstract. Neural networks are able to learn more patterns with the incremental learning than with the correlative learning. The incremental learning is a method to compose an associate memory using a chaotic neural network. The capacity of the network is found to increase along with its size which is the number of the neurons in the network and to be larger than the one with correlative learning. In former work, the capacity was over the direct proportion to the network size with suitable pairs of the refractory parameter and the learning parameter. In this paper, the refractory parameter and the learning parameter are investigated through the computer simulations changing these parameters. Through the computer simulations, it turns out that the appropriate parameters lie near the origin with some relation between them.

1 Introduction

The incremental learning proposed by the authors is highly superior to the autocorrelative learning in the ability of pattern memorization[1,2]. The idea of the incremental learning is from the automatic learning[3]. In the incremental learning, the network keeps receiving the external inputs. If the network has already known an input pattern, it recalls the pattern. Otherwise, each neuron in it learns the pattern gradually. The neurons used in this learning are the chaotic neurons, and their network is the chaotic neural network, which was developed by Aihara[4].

In former work, we investigated the capacity of the networks[5] and the error correction capability[6]. Through the simulations, we found that the capacity is in proportion to the network size with the appropriate parameter which is inverse proportion to the size and that the capability decreases gradually as the number of the learned patterns increases.

In this paper, first, we explain the chaotic neural networks and the incremental learning and refer to the former work on the capacities[7], then the refractory parameter and the learning parameter are investigated, with simulations changing the refractory parameter and the learning parameter counting the capacity of the network in the 100 neuron network.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 1–9, 2013.

2 Chaotic Neural Networks and Incremental Learning

The incremental learning was developed by using the chaotic neurons. The chaotic neurons and the chaotic neural networks were proposed by Aihara[4]. We presented the incremental learning which provided an associative memory[1]. The network type is an interconnected network, in which each neuron receives one external input, and is defined as follows[4]:

$$x_i(t+1) = f\big(\xi_i(t+1) + \eta_i(t+1) + \zeta_i(t+1)\big) \ , \tag{1}$$

$$\xi_i(t+1) = k_s\xi_i(t) + \upsilon A_i(t) \ , \tag{2}$$

$$\eta_i(t+1) = k_m\eta_i(t) + \sum_{j=1}^{n} w_{ij}x_j(t) \ , \tag{3}$$

$$\zeta_i(t+1) = k_r\zeta_i(t) - \alpha x_i(t) - \theta_i(1-k_r) \ , \tag{4}$$

where $x_i(t+1)$ is the output of the i-th neuron at time $t+1$, f is the output sigmoid function described below in (5), k_s, k_m, k_r are the time decay constants, $A_i(t)$ is the input to the i-th neuron at time t, υ is the weight for external inputs, n is the size—the number of the neurons in the network, w_{ij} is the connection weight from the j-th neuron to the i-th neuron, and α is the parameter that specifies the relation between the neuron output and the refractoriness.

$$f(x) = \frac{2}{1+\exp(\frac{-x}{\varepsilon})} - 1 \ . \tag{5}$$

The parameters in the chaotic neurons are assigned in Table 1.

Table 1. Parameters

$\upsilon = 2.0$,
$k_s = 0.95$,
$k_m = 0.1$,
$k_r = 0.95$,
$\theta_i = 0$,
$\varepsilon = 0.015$

In the incremental learning, each pattern is inputted to the network for some fixed steps before moving to the next. In this paper, this term is fixed to 50 steps, and 1 set is defined as a period for which all the patterns are inputted. The patterns are inputted repeatedly for some fixed sets.

During the learning, a neuron which satisfies the condition (6) changes the connection weights as in (7)[1].

$$\xi_i(t) \times (\eta_i(t) + \zeta_i(t)) < 0 \ . \tag{6}$$

$$w_{ij} = \begin{cases} w_{ij} + \Delta w, & \xi_i(t) \times x_j(t) > 0 \\ w_{ij} - \Delta w, & \xi_i(t) \times x_j(t) \leq 0 \end{cases} \quad (i \neq j) \ , \tag{7}$$

where Δw is the learning parameter.

If the network has learned a currently inputted pattern, the mutual interaction $\eta_i(t)$ and the external input $\xi_i(t)$ are both positive or both negative at all the neurons. This means that if the external input and the mutual interaction have different signs at some neurons, a currently inputted pattern has not been learned completely. Therefore, a neuron in this condition changes its connection weights. To make the network memorize the patterns firmly, if the mutual interaction is less than the refractoriness $\zeta_i(t)$ in the absolute value, the neuron also changes its connection weights.

In this learning, the initial values of the connection weights can be 0, because some of the neurons' outputs are changed by their external inputs and this makes the condition establish in some neurons. Therefore, all initial values of the connection weights are set to be 0 in this paper. $\xi_i(0)$, $\eta_i(0)$, and $\zeta_i(0)$ are also set to be 0.

To confirm that the network has learned a pattern after the learning, the pattern is tested on the normal Hopfield's type network which has the same connection weights as the chaotic neural network. That the Hopfield's type network with the connection weights has the pattern in its memory has the same meaning as that the chaotic neural network recalls the pattern quickly when the pattern inputted. Therefore, it is the convenient way to use the Hopfield's type network to check the success of the learning.

3 Capacity

In this section, we retrace the simulations in the former work[7]. In the former works[5,6,7], it turned out to be important for the incremental learning that the connection weights were reinforced by the effect of the refractoriness $\zeta_i(t)$ in the learning condition (6). This suggests that α is significant to the incremental learning. Therefore, we regarded not only Δw but also α as learning parameters, although the other parameters are not meant to have no effect.

First, the refractory parameter α was fixed to be 2.0 and the capacity was inspected changing Δw. The capacity is proportional to its size with the proportional constant 0.92 as shown in Fig. 1. The capacity of the auto-correlative learning is also shown in Fig. 1 and the proportional constant is 0.07.

Under consideration of the former works[5,6], we took $\Delta w = 0.0001$ and $\alpha = 0.1$ and used a 200 neuron network with 250 input patterns. Fig. 2[7] shows the result. The horizontal axis is the learning sets which means learning period and the vertical axis is the number of learned patterns at the end of that sets. The result shows that the capacity of this network is equal to or more than 250 patterns, which means the capacity of the network exceeds the direct proportion to its size.

Next, to investigate the usable pair of Δw and α, the following simulations were carried out. In these simulation, 200 neuron network was used with 240 input patterns and the number of learned patterns was counted changing Δw and α. Because the appropriate value of Δw was 0.005 in former work[5,6], the parameter Δw was changed from 0.0001 to 0.01 in increments of 0.0001 to cover

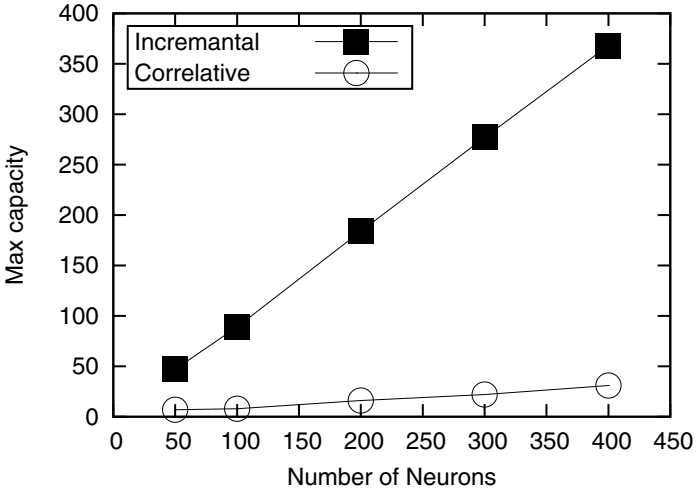


Fig. 1. Capacity of Network at $\alpha = 2.0$

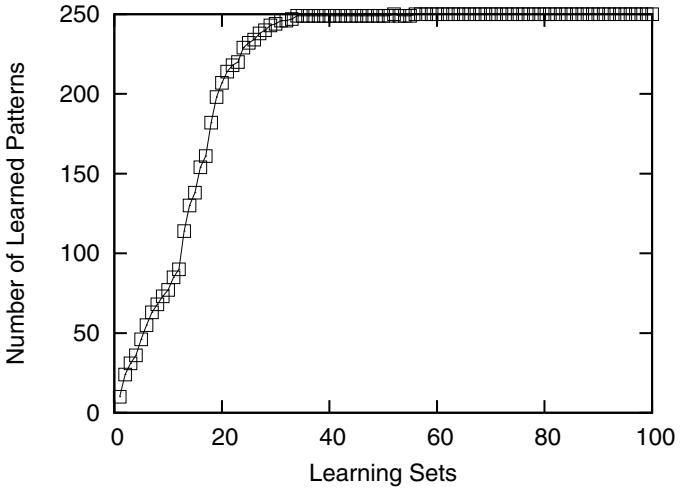


Fig. 2. Number of Learned Patterns in 200 Neuron Network

$\Delta w = 0.005$. The parameter α was changed from 0.01 to 2.00 in increments of 0.01 to cover $\alpha = 0.1$ which was used in Fig 2.

Fig. 3 shows the result of these simulations. From this result, not only Δw but also α strongly affects the number of learned patterns. Around $\Delta w = 0.0011$ and $\alpha = 0.41$, all the 240 input patterns are learned.

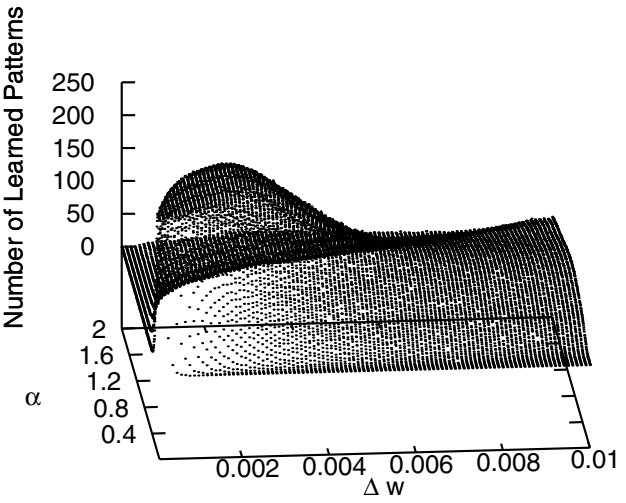


Fig. 3. Number of Success with α and Δw

4 Appropriate Pairs

In this section, we investigate the appropriate pair of Δw and α, with which the network was able to learn all the inputted patterns. To reduce computational complexity, we took the 100 neuron network in this section. The refractory parameter α is changed in increments of 0.01 and the learning parameter Δw is changed in increments of 0.0001.

Through the simulations, the capacity turns out to be 131 with α and Δw listed in Table 2. Fig. 4 shows the result of the simulation with $(\alpha, \Delta w) = (0.02, 0.0001)$. Both axes are the same as the ones in Fig. 2. Because the appropriate pairs listed in Table 2 are located in low area of Δw, the simulations have to be carried out with finer values of Δw in this area.

In the next simulations, the refractory parameter α is changed from 0.001 to 0.2 in increments of 0.001 and the learning parameter Δw is changed from 10^{-6} to 10^{-4} in increments of 10^{-6}. In these simulations, the capacity of the network grew to 145. Fig.5 shows the number of learned patterns when 145 patterns are inputted to the network. The appropriate pairs are plotted with large dots, so that the area of these pairs looks blacker than the other. In Fig.5, the appropriate pairs are lined in an area near $\alpha = 0.01$.

From the result, next simulations are carried out with smaller parameters. In these simulations, the refractory parameter α is changed from 10^{-5} to 10^{-3} in increments of 10^{-5} and the learning parameter Δw is changed from 10^{-7} to 10^{-5} in increments of 10^{-7}. Again, the capacity of the network grew to 149. Fig.6 shows the number of learned patterns when 149 patterns are inputted to the network. The appropriate pairs are lined in an area around $\alpha = 180\Delta w$.

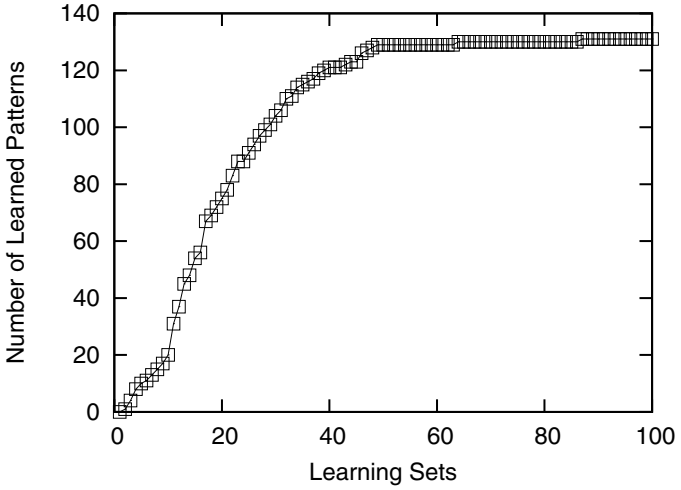


Fig. 4. Number of Learned Patterns in 100 Neuron Network with $\alpha = 0.02$ and $\Delta w = 0.0001$

Table 2. Appropriate α and Δw in the 100 Neuron Network

α	Δw
0.02	0.0001
0.03	0.0002
0.04	0.0002
0.05	0.0003
0.06	0.0003
0.07	0.0004
0.08	0.0005

From above results, the capacity of the network is increasing as the parameters are decreasing. Then, for the third time, the simulations are carried out, with the refractory parameter α changed from 4×10^{-7} to 4×10^{-5} in increments of 4×10^{-7} and with the learning parameter Δw changed from 10^{-9} to 10^{-7} in increments of 10^{-9}.

These simulations revealed that the capacity of the network is still 149. Fig.7 shows the number of learned patterns when 149 patterns are inputted to the network. The appropriate pairs are lined in an area around $\alpha = 193\Delta w$. From the fact that these parameters are two digits smaller than the ones in Fig 6, and that the capacities are the same, 149 can be the maximum capacity of the network.

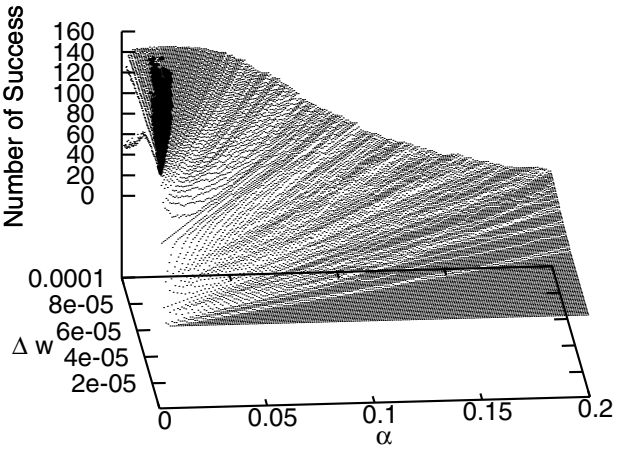


Fig. 5. Number of Learned Patterns in 100 Neuron Network with $\alpha = 0.001$ to 0.2 and $\Delta w = 10^{-6}$ to 10^{-4}

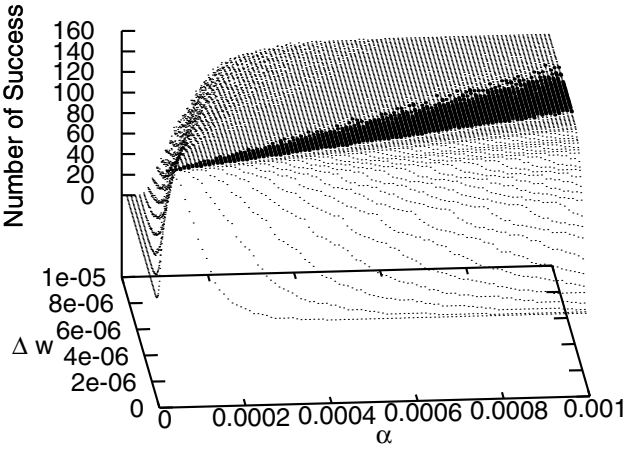


Fig. 6. Number of Learned Patterns in 100 Neuron Network with $\alpha = 10^{-5}$ to 10^{-3} and $\Delta w = 10^{-7}$ to 10^{-5}

For α and Δw, there are appropriate pairs of these parameters, which are in an area around $\alpha \simeq 190\Delta w$ near the origin. We consider that smaller Δw means finer tuning of the connection weights, that smaller α is needed for smaller Δw to keep the balance in the condition (6), and that thus the appropriate pairs lie near the origin. Because the capacity can change with the parameters, to use the maximum capacity of the network, it is important to verify the these parameters which tend to be near the origin with the relation of $\alpha \simeq 190\Delta w$.

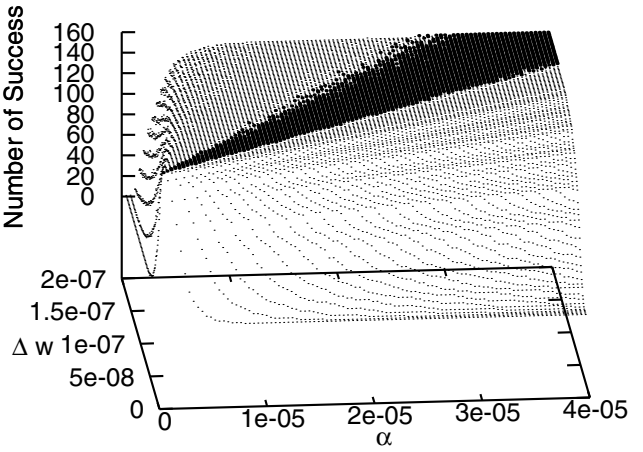


Fig. 7. Number of Learned Patterns in 100 Neuron Network with $\alpha = 4 \times 10^{-7}$ to 4×10^{-5} and $\Delta w = 2 \times 10^{-9}$ to 2×10^{-7}

5 Conclusion

From the former work, the capacity of the network with the incremental learning increases in proportion to its size with the proportional constant 0.92 at the appropriate Δw, and the capacity had exceeded the direct proportion to its size using a suitable α.

In this paper, the refractory parameter and the learning parameter were investigated, counting the capacity of the network. The results shows that the appropriate α and Δw lie near the origin with the relation $\alpha \simeq 190\Delta w$. We just used a 100 neuron network, so there is a possibility that the appropriate α and Δw differ in the other networks, and that is a future work.

References

1. Asakawa, S., Deguchi, T., Ishii, N.: On-Demand Learning in Neural Network. In: Proc. of the ACIS 2nd Intl. Conf. on Software Engineering, Artificial Intelligence, Networking & Parallel/Distributed Computing, pp. 84–89 (2001)
2. Deguchi, T., Ishii, N.: On Refractory Parameter of Chaotic Neurons in Incremental Learning. In: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004, Part II. LNCS (LNAI), vol. 3214, pp. 103–109. Springer, Heidelberg (2004)
3. Watanabe, M., Aihara, K., Kondo, S.: Automatic learning in chaotic neural networks. In: Proc. of 1994 IEEE Symposium on Emerging Technologies and Factory Automation, pp. 245–248 (1994)
4. Aihara, K., Tanabe, T., Toyoda, M.: Chaotic neural networks. Phys. Lett. A 144(6,7), 333–340 (1990)

5. Deguchi, T., Matsuno, K., Ishii, N.: On Capacity of Memory in Chaotic Neural Networks with Incremental Learning. In: Lovrek, I., Howlett, R.J., Jain, L.C. (eds.) KES 2008, Part II. LNCS (LNAI), vol. 5178, pp. 919–925. Springer, Heidelberg (2008)
6. Deguchi, T., Matsuno, K., Kimura, T., Ishii, N.: Error Correction Capability in Chaotic Neural Networks. In: 21st IEEE International Conference on Tools with Artificial Intelligence, Newark, New Jersey, USA, pp. 687–692 (2009)
7. Matsuno, K., Deguchi, T., Ishii, N.: On Influence of Refractory Parameter in Incremental Learning. In: Lee, R. (ed.) Computer and Information Science 2010. SCI, vol. 317, pp. 13–21. Springer, Heidelberg (2010)

Vector Generation and Operations in Neural Networks Computations

Naohiro Ishii[1], Toshinori Deguchi[2], Masashi Kawaguchi[3], and Hiroshi Sasaki[4]

[1] Aichi Institute of Technology, Toyota, Japan
ishii@aitech.ac.jp
[2] Gifu National College of Technology, Gifu, Japan
deguchi@gifu-nct.ac.jp
[3] Suzuka College of Technology, Mie, Japan
masashi@elec.suzukact.ac.jp
[4] Fukui University of Technology, Fukui, Japan
hsasaki@ccmails.fukui-ut.ac.jp

Abstract. To make clear the mechanism of the visual movement is important in the visual system. The problem is how to perceive vectors of the optic flow in the network. First, the biological asymmetric network with nonlinearities is analyzed for generating the vector from the point of the network computations. The results are applicable to the V1 and MT model of the neural networks in the cortex. The stimulus with a mixture distribution is applied to evaluate their network processing ability for the movement direction and its velocity, which generate the vector. Second, it is shown that the vector is emphasized in the MT than the V1. The characterized equation is derived in the network computations, which evaluates the vector properties of processing ability of the network. The movement velocity is derived, which is represented in Wiener kernels. The operations of vectors are shown in the divisive normalization network , which will create curl or divergence vectors in the higher neural network as MST area.

1 Introduction

In the biological neural networks, the sensory information is processed effectively and speedily. Reichard[1] evaluated the sensory information by the auto-correlations in the neural networks. Motion perception is a basic mechanism in the layered visual system of the brain[2]. In this paper, it is shown that the model of the brain cortex proposed by Heeger et al.[4,5] plays an important role in the movement detection and its direction in the neural network computations. First, we analyze the retinal circuit of the catfish[11,12], in which asymmetric sub-network with nonlinearity has a role of the detection of movement of the stimulus by applying Wiener kernel. It is shown that the asymmetric network with nonlinearity detects the movement direction of the stimulus, which has characteristics as a vector. The model of V1 followed by MT in the cortex is represented by the approximated networks. It is shown that the approximated network is closely related to the asymmetric sub-network. Then, the velocity is shown to be emphasized in MT than V1. Beck,J.M et al. derived the activity formula of the

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 10–19, 2013.

divisive normalization circuit[10]. The divisive normalization circuit is included in the model of V1 followed by MT. Then, by applying the activity formula of the divisive normalization circuit in approximated V1 and MT networks, the vector sequence of the optic flow of the stimulus is derived. Thus, it is shown that the operations of the vectors are discussed in the generated vector field of the optic flow.

2 Biological Neural Networks

In the structure of the biological neural networks, neurons are arranged in a systematic way in their structure based on retinal networks. This will be caused for the efficient computation of the collaborative activities. Further, layered structure of the networks shows a prominent feature for the collaborative computations and controls of the total system. Here, we present an example of layered neural network in the biological visual cortex. Fig.1 shows two layered neural network studied by Simoncelli and Heeger [4], which connects networks of V1 followed by MT, where V1 is the front part of the total network, while MT is the rear part of it in Fig.1.

The sub-network model of V1 and MT has nonlinear characteristics followed by a -net normalization sub-network after half-wave rectification circuit.

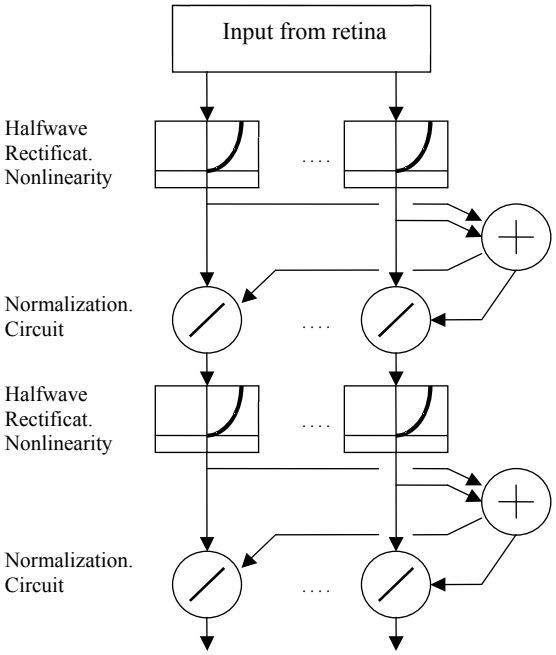


Fig. 1. Model of neural network of brain cortex V1 followed by MT (Simoncelli & Heeger[4])

On the V1 cell and MT cell, velocities are discussed in the visual motion analysis [Fukushima, 2007]. Schematic velocities are shown in Fig. 2, which is shown in Fukushimas V1 and MT neural mode [9]. The directed arrows in V1 receptive field, show the velocity in the horizontal and vertical directions in the left side of Fig. 2, while those in MT receptive field show also larger velocities in the right side of Fig. 2.

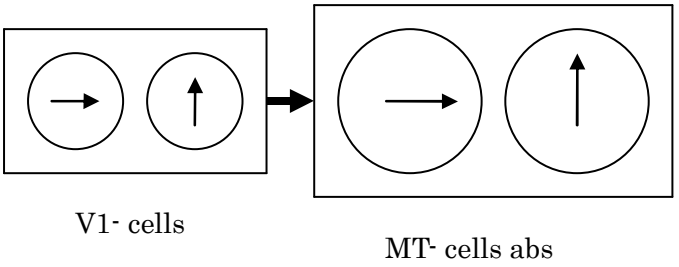


Fig. 2. Schematic vector representation in cells V1 and MT areas

The problem is how to relate the connected V1 and MT neural model in Fig.1 to the vector formation as shown in Fig. 2 in logical formulation. To solve the problem, the connected network is transformed to the approximated network by the following steps

(1) Normalization of a neuron in Fig,1 implies an aspect of saturated activity of neuron responses.

(2) Nonlinearity of the halfwave, squaring rectification and normalization of neuron is approximated as a sigmoid nonlinear function.

(3) The sigmoid nonlinear function is approximated in Taylor series.

(4) The Taylor series expansion is expressed as a transformed network.

3 Analysis Based on Asymmetrical Sub-networks

Respective neurons in the network seem to work independently as they are observed respectively. But, respective neurons work according to the background laws or rules for the total objective function. To make clear functions of the combined approximated network in Fig.1, we applied the research results of the asymmetric sub-network analysis[11,12] and we developed here new formulas for vector computations.. In Fig.3, we assume two kinds of neurons, which are seen in biological retinal network[6,11,12]. One is a linear function neuron and the other is a nonlinear function neuron. The nonlinear neuron works as a nonlinear function of squaring, which show the function of quadratic characteristics. To make clear the function of the combined approximated network in Fig.3, the basic sub-network is analyzed first, since it consists of the basic sub-network in Fig.3.

3.1 Architecture of Asymmetric Sub-networks

Movement perception is carried out firstly in the retinal neural network as the special processing between neurons. The following asymmetric neural network is extracted from the catfish retinal network [6]. The asymmetric structure network with a quadratic nonlinearity is shown in Fig.3, which composes of the pathway from the bipolar cell B1 to the amacrine cell N and that from the bipolar cell B2 , via the amacrine cell N with squaring function to the N cell.

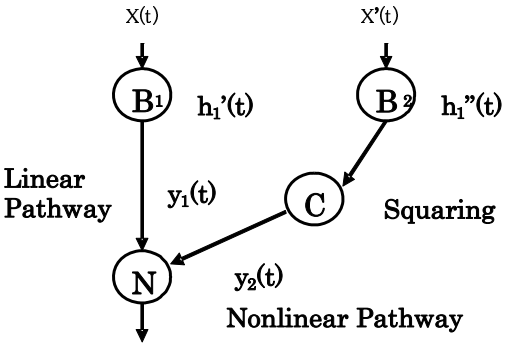


Fig. 3. Asymmetric network with linear and squaring nonlinear pathways

Fig.3 shows a network which plays an important role in the movement perception as the fundamental network. It is shown that N cell response is realized by a linear filter, which is composed of a differentiation filter followed by a low-pass filter. Thus, the asymmetric network in Fig.3 is composed of a linear pathway and a nonlinear pathway. Here, the stimulus with Gaussian distribution is assumed to move from the left side to the right side in the network in Fig.3, as shown in Fig.4. $x''(t)$ is mixed with $x(t)$. Then , we indicate the right stimulus by $x'(t)$. By introducing a mixed ratio , α , the input function of the right stimulus , is described in the following equation , where $0 \leq \alpha \leq 1$ and $\beta = 1 - \alpha$ hold. Then, Fig.4 shows that the moving stimulus is described in the following equation,

$$x'(t) = \alpha x(t) + \beta x''(t) \tag{1}$$

Let the power spectrums of $x(t)$ and $x''(t)$, be p and p' ,respectively an equation $p = k\,p''$ holds for the coefficient k , because we assumed here that the deviations of the input functions are different in their values. Fig.4 shows that the slashed light is moving from the receptive field of B1 cell to the field of the B2 cell .

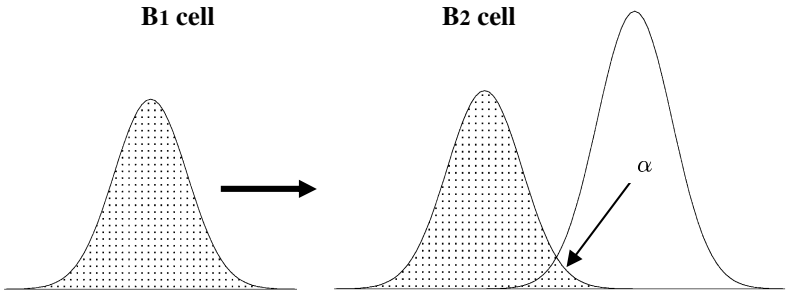


Fig. 4. Stimulus movement from the left to the right side

The mixed ratio of the input $x(t)$, α is shown in the receptive field of B2 cell First , on the linear pathway of the asymmetrical network in Fig.3, the input function is $x(t)$ and the output function is $y(t)$, which is an output after the linear filter of the cell N.

$$y(t) = \int h_1'''(\tau)(y_1(t-\tau) + y_2(t-\tau))d\tau + \varepsilon \tag{2}$$

where $y_1(t)$ shows the linear information on the linear pathway $y_2(t)$ shows the nonlinear information on the nonlinear pathway and ε shows error value. The $y_1(t)$ and $y_2(t)$ are given, respectively as follows,

$$y_1(t) = \int_0^\infty h_1'(\tau)x(t-\tau)d\tau \tag{3}$$

$$y_2(t) = \int_0^\infty \int_0^\infty h_1''(\tau_1)h_1''(\tau_2)x'(t-\tau_1)x'(t-\tau_2)d\tau_1 d\tau_2 \tag{4}$$

We assume here the linear filter N to have only summation operation without in the analysis. Thus the impulse response function $h_1'''(t)$ is assumed to be value 1 without loss of generality.

3.2 Vector Generation in the Sub-networks

Under the assumption that the impulse response functions, $h_1'(t)$ of the cell B1, $h_1''(t)$ of the cell B2 and moving stimulus ratio α in the right to be unknown, the optimization of the network is carried out. By the minimization of the mean squared value ξ of ε in equation (4), the following necessary equations for the optimization of equations are derived,

$$\frac{\partial \xi}{\partial h_1'(t)} = 0 \quad , \quad \frac{\partial \xi}{\partial h_2''(t)} = 0 \quad , \quad \frac{\partial \xi}{\partial \alpha} = 0 \tag{5}$$

Then, the following three equations are derived for the optimization satisfying the equation (5).

$$\begin{aligned} E\left[y(t)x(t-\lambda)\right] &= h_1'(\lambda)p \\ E\left[(y(t)-C_0)x(t-\lambda_1)x(t-\lambda_2)\right] &= 2p^2\alpha^2 h_1''(\lambda_1)h_1''(\lambda_2) \\ E\left[(y(t)-C_0)x'(t-\lambda_1)x'(t-\lambda_2)\right] &= 2p'^2 h_1''(\lambda_1)h_1''(\lambda_2) \end{aligned} \tag{6}$$

where C_0 is the mean value of, $y(t)$ which is shown in the following. Here, the equations (6) can be rewritten by applying Wiener kernels, which are related with input and output correlations method developed by Lee and Schetzen[7]. From the necessary optimization equations in (5), the following Wiener kernel equations are derived as shown in the following[7]. First, we can compute the 0-th order Wiener kernel C_0 , the 1-st order one and $C_{11}(\lambda)$ the 2-nd order one $C_{21}(\lambda_1,\lambda_2)$ on the linear pathway by the cross-correlations between $x(t)$ and. $y(t)$. The suffix i , j of the kernel, $C_{ij}(\bullet)$ shows that i is the order of the kernel and $j = 1$ means the linear pathway, while $j = 2$ means the nonlinear pathway. Then , the 0-th order kernel under the condition of the spatial interaction of cell's impulse response functions $h_1'(t)$ and $h_1''(t)$ becomes

$$C_{11}(\lambda) = \frac{1}{p} E[y(t)x(t-\lambda)] = h_1'(\lambda) \tag{7}$$

since the last term of the second equation becomes zero. The 2-nd order kernel is also derived from the optimization equation as follows,

$$\begin{aligned} C_{21}(\lambda_1,\lambda_2) &= \frac{1}{2p^2} E[(y(t)-C_0)(x(t-\lambda_1)x(t-\lambda_2)] \\ &= \alpha^2 h_1''(\lambda_1)h_1''(\lambda_2) \end{aligned} \tag{8}$$

From equations (7) and (8) , the ratio , α which is a mixed coefficient of $x(t)$ to, is $x'(t)$ shown by α^2 as the amplitude of the second order Wiener kernel. Second , on the nonlinear pathway , we can compute the 0-th order kernel , C_0 the 1-st order kernel $C_{12}(\lambda)$ and the 2-nd order kernel by the $C_{22}(\lambda_1,\lambda_2)$ cross-correlations between $x(t)$ and $y(t)$ as shown in the following, which are also derived from the optimization equations.

$$\begin{aligned} C_{12}(\lambda) &= \frac{1}{p(\alpha^2 + k\beta^2)} E[y(t)x'(t-\lambda)] \\ &= \frac{\alpha}{\alpha^2 + k(1-\alpha)^2} h_1'(\lambda) \end{aligned} \tag{9}$$

and

$$C_{22}(\lambda_1, \lambda_2) = h_1''(\lambda_1) h_1''(\lambda_2) \tag{10}$$

The motion problem is how to detect the movement in the increase of the ratio α in Fig.4. This implies that for the motion of the light from the left side circle to the right one, the ratio α can be derived from the kernels described in the above, in which the second order kernels C_{21} and C_{22} are abbreviated in the representation of equations (11) and (12).

$$(C_{21}/C_{22}) = \alpha^2 \tag{11}$$

holds. Then , from the equation (11) the ratio α is shown as follows

$$\alpha = \sqrt{\frac{C_{21}}{C_{22}}} \tag{12}$$

The equation (12) is called here α - equation , which implies the movement stimulus on the network and shows the detection of the movement by the α without it's direction. This shows that the α - equation is determined by the second order kernels on the linear pathway and the nonlinear one in the network. From the first order kernels C_{11} and C_{12} , and the second order kernels in the above derivations, the movement equation from the left to the right, holds as shown in the following,

$$\frac{C_{12}}{C_{11}} = \frac{\sqrt{\frac{C_{21}}{C_{22}}}}{\frac{C_{21}}{C_{22}} + k\left(1 - \sqrt{\frac{C_{21}}{C_{22}}}\right)^2} \tag{13}$$

It can be shown that the movement equation for the null direction from the right to the left, is derived similarly , which is different from the equation (13) under the condition of the parameter $0 < \alpha < 1$. Thus, the equation (13) shows the direction of the moving stimulus from the left to the right. Next problem is what kinds of functions are needed in the neural networks to realize the movement equations, which are equations (12) and (13). The equations (12) and (13) are derived by computing Wiener nonlinear analysis, which is based on the temporal correlations. It is suggested that some correlation computations will play an important role in the biological neural networks as shown in the following. From equation(8), the velocity $v(t) = \frac{d\alpha}{dt}$ is derived as the following equation (14),

$$\frac{dC_{21}(\tau_1, \tau_2)}{dt} = 2\alpha \frac{d\alpha}{dt} h_1''(\tau_1) h_1''(\tau_2) = 2\alpha v(t) h_1''(\tau_1) h_1''(\tau_2)$$

$$v(t) = \frac{1}{2\sqrt{C_{21}C_{22}}} \frac{dC_{21}(\tau_1, \tau_2)}{dt} \tag{14}$$

4 Operations in Vector Field of Normalization Network

Beck, J.M. et al.[10] derived a new aspect of the divisive normalization circuit, which is called the marginalization in neural circuit with divisive normalization. This normalization circuit is proposed in V1-MT network model by Heeger et al.[4,5] as shown in Fig.1. This normalization circuit is also applicable to the model of MST, which is followed by MT for creating characteristic vector operations as curl or divergence. Beck,J.M. et al.[10] show that biologically plausible networks can implement marginalization near optimally for coordinate transformations, object tracking, simplified olfaction, and causal reasoning. The networks are relatively multilayer recurrent networks that implements a quadratic nonlinearity and divisive normalization. Population code is introduced to determine how neurons encode the likelihood functions and probability distributions from the Bayes approach. The marginalization of the coordinate transformation theory is proposed. Then, the variance of the posterior, σ^2, is inversely proportional to the gain of the activity. Further, on the coordinate transformation, the neural activity A for the one task and another neural activity B for the other task, are assumed. By the marginalization of the neural activities, A and B, the integrated neural activity is generated as the C. Then, by the linear coordinate transformation, the integrated variance is shown as $\sigma_C^2 = \sigma_A^2 + \sigma_B^2$. This shows that the integrated gain of the neural activity becomes $1/g^C = 1/g^A + 1/g^B$, which may be written as

$$g^C = \frac{g^A g^B}{g^A + g^B} \tag{15}$$

Thus, the gains transform via a quadratic nonlinearity with divisive normalization. The response activity, which is proportional to the gain in equation (15), is replaced as [10]

$$r_k^C = \frac{\sum_{ij} w_{ij}^k r_i^A r_j^B}{\sum_l (\alpha_l^A r_l^A + \beta_l^B r_l^B)} \tag{16}$$

where the $w's$, $\alpha's$ and $\beta's$ are coefficient weights. Here, r_i^A and r_j^B correspond to activity in the input layer and r_k^C to activity in the output layer. The equation (16) is useful for the vector operations for the detection and estimation of differential forms in optic flows, which also shows a quadratic nonlinearity with divisive normalization.

The suffix of A, B and C in the equation (16) can be interpreted as the vectors A , B and C..

From the equation (16), the vector dot product in the vector operation, is introduced.

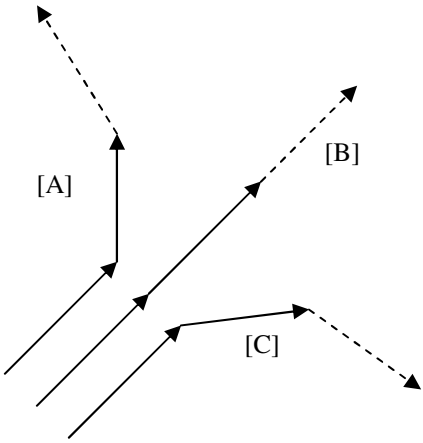


Fig. 5. A partial representation of simplified optic flow vectors

A simplified example of the optic flow is shown in Fig.5, in which one arrow shows a vector of the optic flow. The vector flow [A] shows a curved vectors sequence to the left side as shown in Fig.5. Also, the vector flow [B] to the right side and the vector flow [C] diverges straightly. The vector flow [A] is characterized by applying the equation(16).

The Stokes' theorem is computed by the line integral of vectors, which is essentially calculated by the vector dot product, which will be carried out in the marginalization circuit. The estimated vector sequence points the coordinates of the vectors sequence. By using these coordinate values, the line integral of the Stokes' theorem is computed. The curl vectors shown in MST will be created by the Stokes' theorem. When the counter clockwise rotation of the curved optic flow vectors is taken, the Stokes' theorem shows a negative value. Thus, the line integral of the estimated vector sequence has a negative value. Then, the counter clockwise rotation vector will be made in the MST. When the clockwise rotation of the curved optic flow vectors is taken, the theorem shows a positive value. Then, the clockwise rotation vectors will be created.

5 Conclusion

The neural networks are analyzed to make clear functions of the layered network of V1 followed by MT in the brain cortical area. In this paper, the structure and function of the nonlinear biological asymmetrical network, are analyzed first to detect

the movement direction and its speed of the stimulus from the point of the neural computation. The conditions of the movement and its direction of the stimulus, are derived from these analyses. We applied these results by using Wiener analysis to the connected V1 and MT networks. Then, it was shown that the structure of the approximated V1 and MT network has a higher detection ability for the movement. Finally, the operations of the vectors of optic flow are discussed based on the theory of the divisive normalization circuit, which is derived by Beck,J.M., et al. The vector sequence is applicable to the Stokes' theorem on the vector field. Thus, the curl or divergence vectors will be generated on the higher network as MST, which is followed by MT in the brain.

References

1. Reichard, W.: Autocorrelation, A principle for the evaluation of sensory information by the central nervous system, Rosenblith edn. Wiley, NY (1961)
2. Chubb, C., Sperling, G.: Drift-balanced random stimuli, A general basis for studying non-Fourier motion. J. Optical Soc. of America A, 1986–2006 (1988)
3. Taub, E., Victor, J.D., Conte, M.: Nonlinear preprocessing in short-range motion. Vision Research 37, 1459–1477 (1997)
4. Simonceli, E.P., Heeger, D.J.: A Model of Neuronal Responses in Visual Area MT. Vision Research 38, 743–761 (1996)
5. Heeger, D.J.: Normalization of cell responses in cat striate cortex. Visual Neuroscience 9, 181–197 (1992)
6. Naka, K.-I., Sakai, H.M., Ishii, N.: Generation of transformation of second order nonlinearity in catfish retina. Annals of Biomed. Eng. 16, 53–64 (1988)
7. Lee, Y.W., Schetzen, M.: Measurements of the Wiener kernels of a nonlinear by cross-correlation. Int. J.of Control 2, 237–254 (1965)
8. Sejnowski, T., Koch, C., Churchland, P.: Computational Neuroscience. Science, New Series 241(4871), 1299–1306 (1988)
9. Fukushima, K.: Visual Motion Analysis by a Neural Network. Neural Information Processing 11(4-6), 63–73 (2007)
10. Beck, J.M., Latham, P.E., Pouget, A.: Marginalization in Neural Circuits with Divisive Normalization. Journal of Neuroscience 31(43), 15310–15319 (2011)
11. Ishii, N., Ozaki, M., Sasaki, H.: Correlation Computations for Movement Detection in Neural Networks. In: Negoita, M.G., Howlett, R.J., Jain, L.C. (eds.) KES 2004, Part II. LNCS (LNAI), vol. 3214, pp. 124–130. Springer, Heidelberg (2004)
12. Ishii, N., Deguchi, T., Kawaguchi, M.: Neural Computations by Asymmetric Networks with Nonlinearities. In: Beliczynski, B., Dzielinski, A., Iwanowski, M., Ribeiro, B. (eds.) ICANNGA 2007, Part II. LNCS, vol. 4432, pp. 37–45. Springer, Heidelberg (2007)

Synaptic Scaling Balances Learning in a Spiking Model of Neocortex

Mark Rowan[1] and Samuel Neymotin[2]

[1] School of Computer Science, University of Birmingham, UK
m.s.rowan@cs.bham.ac.uk
[2] Dept. Neurobiology, Yale University School of Medicine, New Haven, USA
samuel.neymotin@yale.edu

Abstract. Learning in the brain requires complementary mechanisms: potentiation and activity-dependent homeostatic scaling. We introduce synaptic scaling to a biologically-realistic spiking model of neocortex which can learn changes in oscillatory rhythms using STDP, and show that scaling is necessary to balance both positive and negative changes in input from potentiation and atrophy. We discuss some of the issues that arise when considering synaptic scaling in such a model, and show that scaling regulates activity whilst allowing learning to remain unaltered.

1 Introduction

Spike Timing-Dependent Plasticity (STDP), a phenomenological learning rule in which synaptic potentiation and depression depend upon relative firing times [1, 2], has been used to learn oscillatory rhythms in neocortical models. In an existing biologically-realistic spiking model of neocortex [3], applying excitatory to excitatory (E→E) STDP with a rhythmic training signal led to hyper-potentiation through positive feedback: strengthened synapses drove postsynaptic neurons to fire immediately, leading to further potentiation. This unbounded potentiation then pushed the network into synchronized epileptiform firing. Directly opposing E→E learning with equal excitatory to inhibitory (E→I) potentiation partially balanced this positive feedback. However, epileptiform behaviour still occurred with high-frequency signals [4].

We postulated that a homeostatic mechanism might be a solution to this problem. Neuronal homeostatic synaptic scaling is a local feedback mechanism which senses levels of activity-dependent cytosolic calcium within the cell and adjusts neuronal firing activity accordingly. This is achieved by producing alterations in excitatory AMPA receptor accumulation in response to changes in firing activity occurring over hours to days [5], leading to changes in the excitability of the neuron.

During learning, synaptic scaling plays an important role in balancing potentiation. By constantly shifting mean activation back towards a target activity level, but maintaining the learned relative distribution of presynaptic weights, global levels of activity can be regulated [6]. During periods of hypoactivity (e.g.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 20–29, 2013.

in degenerative disorders), synaptic scaling is also capable of raising the sensitivity of neurons via AMPA receptor upregulation, so that activity levels can be restored [5].

Previous work has demonstrated synaptic scaling with learning in a single-neuron model [6]. It has also been shown that synaptic scaling can prevent input saturation in a spiking neural network in the absence of learning [7]. In this paper, we add long-term synaptic plasticity to a spiking neural network to show that homeostatic synaptic scaling can prevent hyper-potentiation while preserving learned information.

2 Methods

The model was based on the anatomy of a single column of sensory neocortex [3, 8, 9]. It was composed of 470 neurons divided into 3 types (excitatory pyramidal cells E, fast-spiking interneurons I, and low-threshold spiking interneurons IL), distributed across the 6 layers of the neocortex. This yielded 13 neuronal populations in total, with the following numbers of cells per type: E2 (i.e. excitatory layer 2/3 cell), 150; I2 (fast spiking interneuron in layer 2/3), 25; I2L (low-threshold spiking interneuron in layer 2/3), 13; E4, 30; I4, 20; I4L, 14; E5a, 65; E5b, 17; I5, 25; I5L, 13; E6, 60; I6, 25; and I6L, 13.

The cell model was an extension of an integrate-and-fire unit with added complexity (adaptation, bursting, depolarization blockade, and voltage-sensitive NMDA conductance) in the form of rules [10], and was simulated in an event-driven fashion where cell state variables were only calculated at input events, making use of previously developed just-in-time synapses optimized for large networks supporting high-frequency synaptic events [11]. Each cell had fast inhibitory GABA_A receptors, fast excitatory AMPA receptors, and slow excitatory NMDA receptors, with each producing a voltage-step with following decay.

In addition to spikes generated by cells in the model, subthreshold Poisson-distributed spike inputs to each synapse were used to maintain activity in the model: 100–150 Hz for GABA_A, 240–360 Hz for AMPA receptors, and 40–60 Hz for NMDA receptors. These external inputs represented the inputs from other regions of the brain. To simulate additional afferent sensory inputs, low-amplitude training signals were applied to the layer 4 excitatory neurons (E4) in some simulations. STDP was implemented on AMPA synapses from E→E cells using a basic model with bilateral exponential decay (40ms maximal interspike difference, 10ms time constant) incrementing by 0.1% of baseline synaptic weight. It should be noted that STDP in this model is additive, whilst van Rossum argues that it should be multiplicative [6]. Further details of the cell model can be found in [3] and [4].

Scaling was implemented at E cell AMPA synapses by multiplying each cell i's postsynaptic input by a scale factor w_i, representing the multiplicative accumulation of AMPA receptors at synapses. Changes in the scale factor were calculated following the formula of van Rossum et al. [6], with a_i as the cell's firing activity, a_i^{goal} as the target activity, β as the scaling strength, γ as the

"integral controller" weight, and $\frac{dw_i(t)}{dt}$ as the rate of change of the synaptic weight:

$$\frac{dw_i(t)}{dt} = \beta w_i(t)[a_i^{goal} - a_i(t)] + \gamma w_i(t) \int_0^t dt'[a_i^{goal} - a_i(t')] \tag{1}$$

The following parameter values were used: strength $\beta = 4.0 \times 10^{-8}$/ms/Hz; integral controller weight $\gamma = 1.0 \times 10^{-10}$/ms^2/Hz; activity sensor time constant $\tau = 100 \times 10^3$ ms. Scaling was applied inversely at GABA$_A$ synapses (i.e. by multiplying postsynaptic input by $\frac{1}{w_i}$) to enable the scaling of excitatory and inhibitory synapses in opposite directions, mimicking the effect of global growth factors such as BDNF [5, 7, 12, 13].

Average activity level for each cell i was sensed using van Rossum's slow-varying sensor $a_i(t)$, which increased monotonically with spike t_x at current timestep t, and decayed otherwise [6]:

$$\tau \frac{da_i(t)}{dt} = -a_i(t) + \sum_x \delta(t - t_x) \tag{2}$$

The sensor decays exponentially as it is updated at each non-firing timestep. However, the use of event-driven just-in-time synapses [4, 11] meant that cell states were only updated upon each spike event rather than at every timestep, so inter-spike decay of the activity sensor could only be calculated periodically. We therefore modified the activity sensor. Here, the first term decays the sensor according to the time between spikes $t - t_x$, and the second term increments it for the new spike, with both terms updated concurrently on the occurrence of a spike at time t_x:

$$a_i(t) = a_i(t_x)e^{-\frac{1}{\tau}(t-t_x)} + \frac{1 - a_i(t_x)}{\tau} \tag{3}$$

Figure 1 shows the activity of a simulated, randomly-spiking neuron operating under the constant-timestep update policy (2), and the equivalent activity values under the periodic-update policy (3). The activity rises identically in both cases when spikes occur, but the periodic sensor does not decay until the next spike event occurs, giving the step-like appearance. The values at the spike times are correct down to round-off error at the spike times.

Instead of providing an arbitrary rate target for each cell, which would fundamentally affect network dynamics, the intrinsic dynamics of the network were used to provide set-points. Initially, with synaptic scaling off, activity sensors began at 0 Hz. They were then adjusted over 800 s of simulated time based on the activity level of the cells. Synaptic scaling was then switched on.

A time constant τ of 100 s [6] leads to a simulation timescale of several hours for synaptic scaling: far closer to the expected biological timescale than previous studies [5, 14, 7]. To achieve this length of simulation, the model was extended to allow periodic flushing of all spike data to disk, enabling very long runs (unlimited except for available disk space). A typical simulation of 44 hours ran in approximately real time and produced around 2 GB of spike

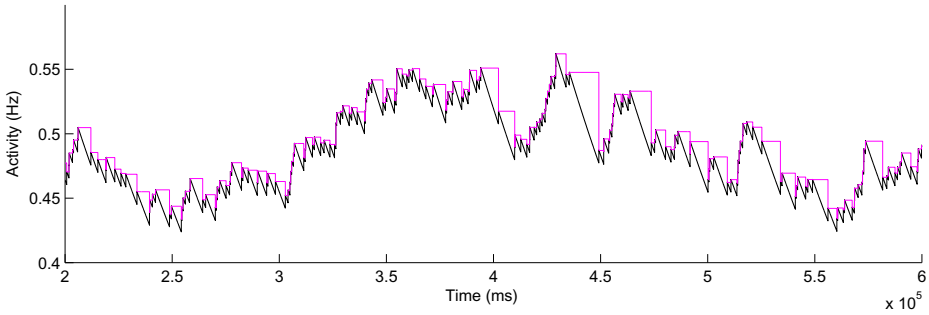


Fig. 1. Activity sensor updating at every simulation timestep (Eqn. 2; black) and at every spike for activity-driven just-in-time synapses (Eqn. 3; magenta).

data. The model was implemented in NEURON 7.2 [15] for Linux, and is available on ModelDB at the following URL: `https://senselab.med.yale.edu/modeldb/enterCode.asp?model=147141`.

Data Analysis. Simulation spike-trains were organized into multiunit activity (MUA) vectors, defined for a cell population as the number of spikes in the population over a time interval (bin). Bin sizes were set to 5 ms (200 Hz sampling rate). Analyses were performed using mean-subtracted MUA vectors, with spectra calculated by the multitaper spectral power estimation method, as implemented in the Python wrapper of the FORTRAN MTSpec library [16].

3 Simulation Results

3.1 Scaling Prolonged Activity during Deletion

In an initial experiment, we demonstrated the usefulness of synaptic scaling by altering network dynamics through gradual removal (pruning) of cells (Fig. 2). Every 1600 s, three I or E neurons were selected at random and removed from the network by setting all their synaptic weights to zero. The global external input weights were scaled down proportionally to the amount of deletion, at a quarter of the deletion rate, to prevent the external inputs from swamping internal activation and artificially raising activity. By the end of the simulation, approximately two thirds of the cells in the network had been deleted.

In the absence of scaling, average firing across E cells declined steadily as cell deletion progressed (Fig. 2 green / lower line). With scaling present, firing activity was maintained (Fig. 2 blue / upper line), with brief activity peaks caused when the inherent delay in the activity sensor led to over-compensation. These activity peaks do not correspond to deletion times, but rather to emergent instabilities in the resulting damaged network. Indeed, the network remains stable for nearly half a day following the onset of deletion after 800 s. The over-compensation can be adjusted to some degree, although not eliminated, by altering the scaling parameters β and γ (not shown).

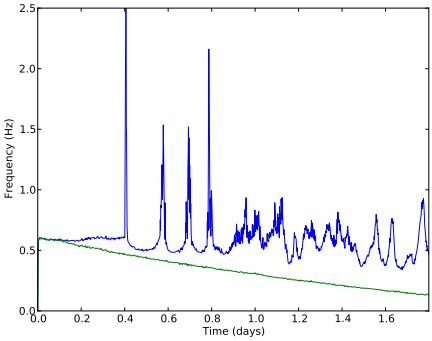
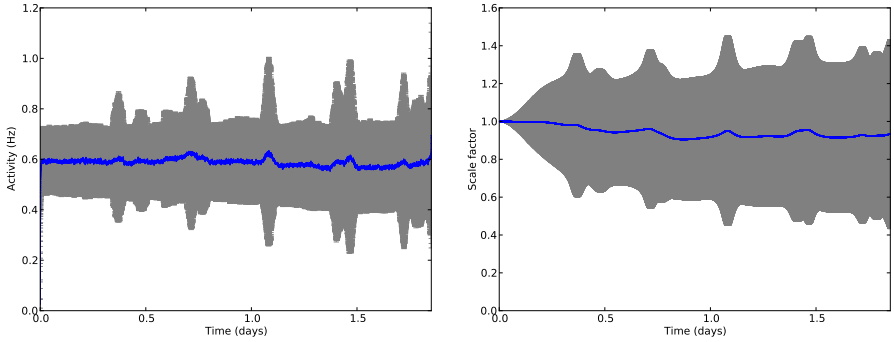


Fig. 2. E activity during pruning with (blue / upper) and without (green / lower) compensatory synaptic scaling. Run time 160,000 s (≈ 44 h).



(a) E activity during scaling (b) Scale factors of E cells

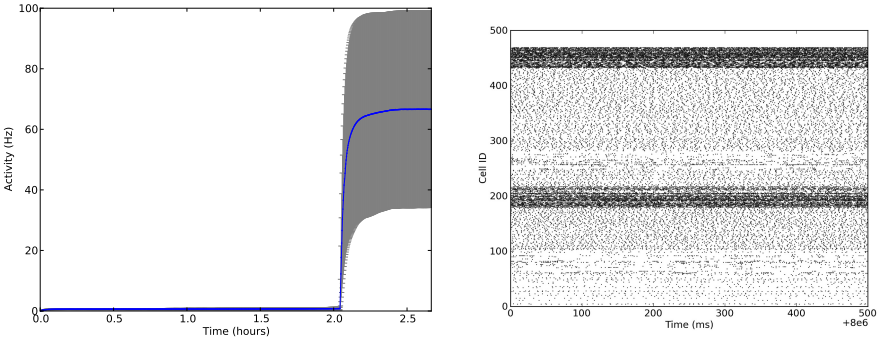
Fig. 3. Scaling does not destabilize the network (mean: blue, std: grey)

3.2 Synaptic Scaling Did Not Disrupt Network Behavior

The model was run for 160,000 s (≈ 44 h) to examine the effects of scaling over time on network dynamics. With scaling, activity of the E cells remained steady (Fig. 3a), and scale factors remained centered around 1 (Fig. 3b). Scaling appeared to preserve stability of the network during these extremely long runs.

3.3 Unrestrained STDP Led to Hyper-potentiation

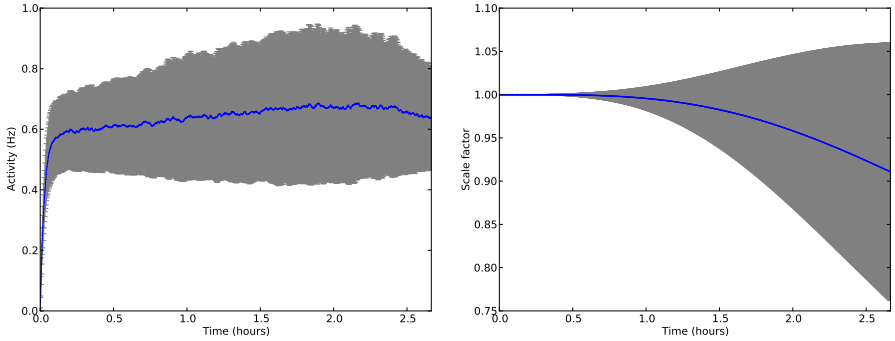
We trained the network by applying a signal consisting of low-weight single spikes at 8 Hz to E4 cells for 8000 s (≈ 2.2 h) in the absence of synaptic homeostasis (Fig. 4). STDP was turned off during the final 800 s in order to test recall. We found that any training signal frequency eventually pushed the network into a state of excessive firing. This occurred even when E→I STDP balancing was added (not shown).



(a) E activity during training

(b) Raster plot of 500 ms showing high-frequency network activity after 2 h

Fig. 4. Training with E→E STDP pushes network to high frequency activity



(a) E activity during training with scaling

(b) Scale factors of E cells

Fig. 5. Synaptic scaling maintains E activity profile during STDP

3.4 Synaptic Scaling Prevented Overactivation

We then assessed the model with STDP, training and synaptic scaling (Fig. 5). Local E cell homeostatic scaling balanced the potentiation caused by STDP, gradually scaling down all E cells, and preventing pathological over-activation.

3.5 Synaptic Scaling Preserved Learning

Synaptic scaling served to maintain cell firing near the target rate, here the baseline rate. However, it was possible that the scaling-down of activity would simply reverse the potentiation caused by STDP, resulting in a loss of learned information. In order to determine whether scaling allowed the learning of oscillations to persist, the power spectra of the E cells were obtained at various points during the learning process using the multitaper spectral analysis method, with spikes

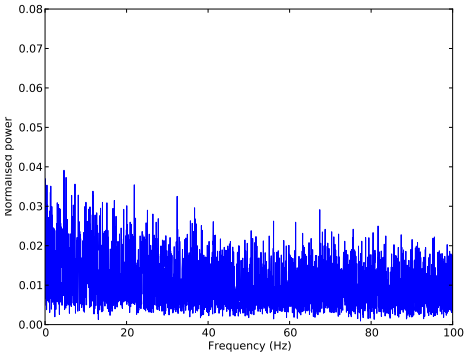


Fig. 6. Baseline power spectrum of E cells (frequency (Hz) vs normalized power)

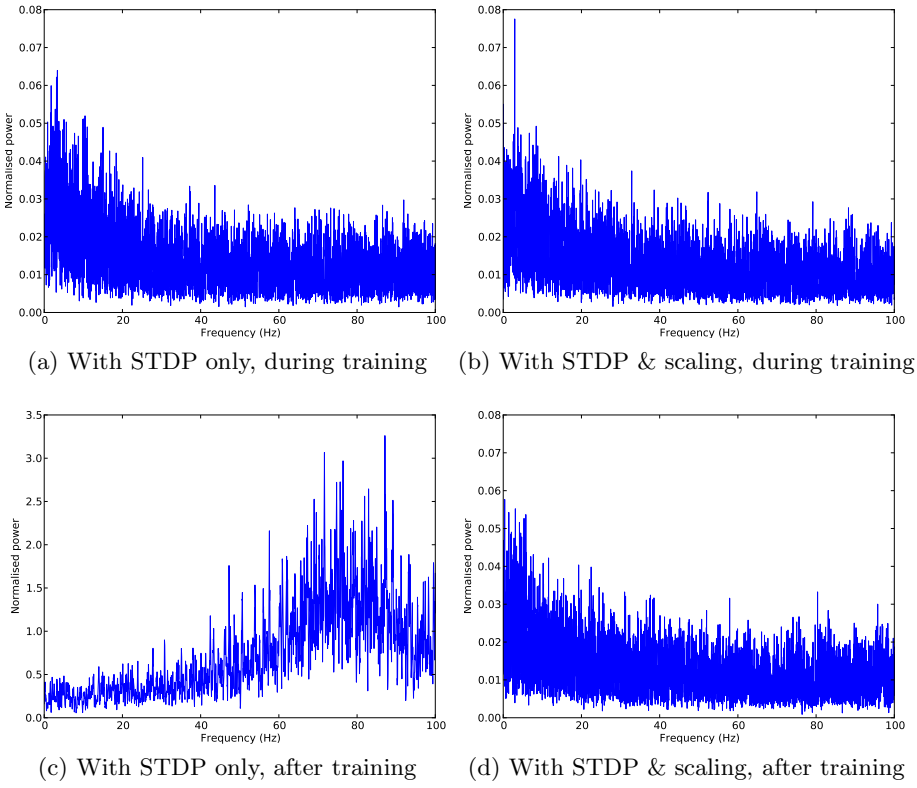
sorted into 5 ms bins (Fig. 6) [16]. These plots show unsmoothed normalised power of the E cells within the network at each of a range of frequencies from 0-100 Hz.

STDP was applied at E→E synapses for 8000 s (≈ 2.2 h) with an 8 Hz sensory signal (Fig. 7d). In one simulation, synaptic scaling was also switched on for E cells. Power spectra were obtained for the period from 5600-6400 s, shortly after the middle of training (Figs. 7a and 7b), and again during the recall period at the end of learning (Figs. 7c and 7d).

In both simulations, it can be seen that STDP has caused a shift in the power spectra, with an increase in the amplitude of oscillations at low frequencies from 0-10 Hz and a decrease above 10 Hz (Figs. 7a and 7b). This demonstrates that the network has learned from the training signal. Shortly after 7400 s (2 h), the network without synaptic scaling transitioned to high-frequency activity, without retention of the 8 Hz training signal (Fig. 7c; note different scale). However, in the network with synaptic scaling turned on, lower frequency activity was maintained, with a peak near the 8 Hz that was imposed during training (Fig. 7d). Synaptic scaling therefore prevented over-activation and preserved learning.

4 Discussion

This research has introduced homeostatic synaptic scaling with dynamically-obtained target activity rates to a realistic spiking model of neocortex which learned oscillatory frequencies via STDP. We demonstrated that scaling is necessary for upregulation of neural activity during decline in input. This might have implications for neurodegenerative brain disorders, in which cortical activation might be expected to decrease. Peaks of activity were observed during deletion due to periodic over-compensation by the scaling mechanism. Experimental observations demonstrating hyperactivity in cells near beta-amyloid plaques in Alzheimer's disease, and the increased incidence of seizures in Alzheimer's patients, suggests these activity peaks may have a biological basis [17, 14, 18]. Additionally, synaptic scaling may play a significant role in the progression of



(a) With STDP only, during training (b) With STDP & scaling, during training

(c) With STDP only, after training (d) With STDP & scaling, after training

Fig. 7. Power spectra during (top)/after (bottom) STDP, with (R)/without (L) scaling

Alzheimer's disease [19–21], and further understanding of this mechanism and its relationship to learning and disease pathology may be crucial to finding better treatments.

We also showed that scaling does not negatively affect the network at baseline, but that it is stable. We demonstrated that E→E scaling is sufficient to balance the hyper-potentiation caused by unrestrained STDP. Potentiation strengthens the co-incident connections between neurons in a positive feedback cycle, eventually leading to hyper-potentiation, but scaling acts to shift the mean activation constantly back towards the target activity. At the same time, the relative (learned) distribution between postsynaptic weights remains unaltered by scaling, and we subsequently demonstrated this principle by showing that learning of an 8 Hz oscillatory signal is not erased by scaling.

This model investigated training and scaling at E→E synapses between E cells. While there is some evidence of STDP in I cells [22], I cells do not appear to perform scaling, but rather: "homeostatic regulation of inhibition is a noncell-autonomous process that either requires changes in both pre-and postsynaptic

activity simultaneously or is triggered by global changes in network activity" (Turrigiano et al., 2011 [13]). In our model, directly enabling synaptic scaling in I cells was found to lead to dramatic instabilities in the network dynamics (even when operating the network at baseline, i.e. without STDP or a sensory signal), which is consistent with Turrigiano's observations. Rather, the network appears to be most stable when I cells are allowed to adjust their activity passively according to the changing output from neighboring E cells, thus requiring only one dimension for the E/I balance rather than needing a second simultaneously active dimension for scaling.

STDP was implemented using an incremental step of 0.1% of baseline synaptic weight, which may seem low. Increasing this step size, however, meant that short bursts of high-frequency activity were seen during learning, as the activity sensors could not respond quickly enough to cause sufficient compensatory scaling (although the network did soon scale back to previous firing rates). However, 8000 s (2 h) of sustained training may also be very long compared to biological learning from hippocampal backprojections, which is known to include periods of recall and consolidation between periods of learning [23]. This would make an interesting avenue for future research.

Acknowledgements. Research funded by EPSRC, DARPA grant N66001-10-C-2008, NIH grant R01MH086638. The authors would like to thank John Bullinaria (Birmingham) and William Lytton (SUNY Downstate) for their helpful comments; Michael Hines and Ted Carnevale (Yale) for NEURON simulator support; Tom Morse (Yale) for ModelDB support; and the anonymous reviewers for their constructive feedback.

References

1. Dan, Y., Poo, M.: Spike timing-dependent plasticity of neural circuits. Neuron 44(1), 23–30 (2004)
2. Zhang, L., Tao, H., Holt, C., Harris, W., Poo, M.: A critical window for cooperation and competition among developing retinotectal synapses. Nature 395(6697), 37–44 (1998)
3. Neymotin, S., Lee, H., Park, E., Fenton, A., Lytton, W.: Emergence of physiological oscillation frequencies in a computer model of neocortex. Front. Comput. Neurosci. 5 (2011)
4. Neymotin, S., Kerr, C., Francis, J., Lytton, W.: Training oscillatory dynamics with spike-timing-dependent plasticity in a computer model of neocortex. In: Signal Processing in Medicine and Biology Symposium (SPMB), pp. 1–6. IEEE (2011)
5. Turrigiano, G.: The self-tuning neuron: synaptic scaling of excitatory synapses. Cell 135(3), 422–435 (2008)
6. Van Rossum, M., Bi, G., Turrigiano, G.: Stable Hebbian learning from spike timing-dependent plasticity. J. Neurosci. 20(23), 8812 (2000)
7. Chandler, B., Grossberg, S.: Joining distributed pattern processing and homeostatic plasticity in recurrent on-center off-surround shunting networks: Noise, saturation, short-term memory, synaptic scaling, and BDNF. Neural Networks (2012)

8. Binzegger, T., Douglas, R., Martin, K.: A quantitative map of the circuit of cat primary visual cortex. The Journal of Neuroscience 24(39), 8441–8453 (2004)
9. Lefort, S., Tomm, C., Floyd Sarria, J., Petersen, C.: The excitatory neuronal network of the C2 barrel column in mouse primary somatosensory cortex. Neuron 61(2), 301 (2009)
10. Lytton, W., Stewart, M.: Rule-based firing for network simulations. Neurocomputing 69(10), 1160–1164 (2006)
11. Lytton, W., Omurtag, A., Neymotin, S., Hines, M.: Just-in-time connectivity for large spiking networks. Neural Comput. 20(11), 2745–2756 (2008)
12. Rutherford, L., Nelson, S., Turrigiano, G.: BDNF has opposite effects on the quantal amplitude of pyramidal neuron and interneuron excitatory synapses. Neuron 21(3), 521–530 (1998)
13. Turrigiano, G.: Too many cooks? intrinsic and synaptic homeostatic mechanisms in cortical circuit refinement. Annu. Rev. Neurosci. 34, 89–103 (2011)
14. Fröhlich, F., Bazhenov, M., Sejnowski, T.: Pathological effect of homeostatic synaptic scaling on network dynamics in diseases of the cortex. The Journal of Neuroscience 28(7), 1709–1720 (2008)
15. Carnevale, N., Hines, M.: The NEURON Book. Cambridge University Press, New York (2006)
16. Prieto, G., Parker, R., Vernon III, F.: A Fortran 90 library for multitaper spectrum analysis. Computers & Geosciences 35(8), 1701–1710 (2009)
17. Busche, M., Eichhoff, G., Adelsberger, H., Abramowski, D., Wiederhold, K., Haass, C., Staufenbiel, M., Konnerth, A., Garaschuk, O.: Clusters of hyperactive neurons near amyloid plaques in a mouse model of Alzheimer's disease. Science Signalling 321(5896), 1686 (2008)
18. Trasande, C., Ramirez, J.: Activity deprivation leads to seizures in hippocampal slice cultures: is epilepsy the consequence of homeostatic plasticity? J. Clin. Neurophysiol. 24(2), 154–164 (2007)
19. Small, D.H.: Network dysfunction in Alzheimer's disease: does synaptic scaling drive disease progression? Trends Mol. Med. 14(3), 103–108 (2008)
20. Rowan, M.: Information-selectivity of beta-amyloid pathology in an associative memory model. Front. Comput. Neurosci. 6(2) (January 2012)
21. Rowan, M.: Effects of Compensation, Connectivity and Tau in a Computational Model of Alzheimer's Disease. In: Proc. IJCNN, pp. 543–550. IEEE (2011)
22. Lamsa, K., Kullmann, D., Woodin, M.: Spike-timing dependent plasticity in inhibitory circuits. Frontiers in Synaptic Neuroscience 2 (2010)
23. McClelland, J., McNaughton, B., O'Reilly, R.: Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory. Psychol. Rev. 102(3), 419–457 (1995)

Can Two Hidden Layers Make a Difference?

Věra Kůrková[1] and Marcello Sanguineti[2]

[1] Institute of Computer Science, Academy of Sciences of the Czech Republic
Pod Vodárenskou věží 2, Prague 8, Czech Republic
vera@cs.cas.cz
[2] DIBRIS - University of Genoa
Via Opera Pia 13, 16145 Genova, Italy
marcello.sanguineti@unige.it

Abstract. Representations of multivariable Boolean functions by one and two-hidden-layer Heaviside perceptron networks are investigated. Sufficient conditions are given for representations with the numbers of network units depending on the input dimension d linearly and polynomially. Functions with such numbers depending on d exponentially or having some weights exponentially large are described in terms of properties of their communication matrices. A mathematical formalization of the concept of "highly-varying functions" is proposed. There is given an example of such function which can be represented by a network with two hidden layers with merely d units.

Keywords: One and two hidden-layer perceptron networks, model complexity, representations of multivariable Boolean functions, communication matrices.

1 Introduction

The most widespread type of a neural network architecture is the one-hidden-layer network. Computational units in the hidden layer are usually perceptrons, radial or kernel units. For regression tasks, one-hidden-layer networks have a single linear output whereas for classification ones, they have a single threshold unit. For one-hidden-layer networks, a variety of learning algorithms have been developed and successfully applied (see, e.g., [1, 2] and the references therein).

One-hidden-layer networks with many types of computational units are known to be universal approximators, i.e., they can approximate up to any desired accuracy all continuous functions and all $\mathcal{L}^p$-functions on compact subsets of $\mathbb{R}^d$. In particular, the universal approximation property holds for one-hidden-layer perceptron networks with non polynomial activation functions [3, 4] and with radial and kernel units satisfying mild conditions [5–7]. Moreover, all functions defined on finite subsets of $\mathbb{R}^d$ can be represented exactly by one-hidden-layer networks with sigmoidal perceptrons [8] or with Gaussian kernel units [9].

Although some proofs of the universal approximation capability of one-hidden-layer networks are constructive, they require potentially unlimited number of hidden units. This number is a critical factor for a practical implementation. Model complexities of one-hidden-layer networks have been studied using tools

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 30–39, 2013.

from nonlinear approximation theory. Estimates of rates of approximation of various classes of multivariable functions by networks with increasing numbers of hidden units were derived (see, e.g., [10] and references therein). Inspection of such upper bounds led to descriptions of families of functions that can be well approximated by one-hidden-layer networks with reasonably small numbers of computational units of various types. On the other hand, limitations of computational capabilities of one-hidden-layer networks are less understood. Only few lower bounds on the approximation error by one-hidden-layer networks are known. Moreover, such bounds are mostly non constructive and hold for types of computational units that are not commonly used [11, 12].

Recently, new learning algorithms, which can be applied to networks with more than one hidden layer, were developed (see, e.g., [13, 14]). Such networks have been called deep networks, in contrast to shallow ones that have merely one hidden layer [13]. As training deep networks involves complicated nonlinear optimization procedures, generally it is more difficult than training shallow ones. Thus it is desirable to develop some theoretical foundations for the characterization of tasks that require considerably larger model complexity and/or size of weights when computed by shallow networks than by deep ones. Since typical applications of neurocomputing deal with large numbers of variables, it is particularly important to understand how quickly model complexities of shallow and deep networks grow with increasing input dimensions.

To contribute to such understanding, we investigate complexity of one and two-hidden-layer Heaviside perceptron networks representating real-valued Boolean functions. These functions occur in applications where input data are represented by binary values. Among Boolean functions, d-dimensional parities received special attention. Several authors (see, e.g., [15, 16]) investigated capabilities of perceptron, SVM and RBF networks to classify d-dimensional data according to their parities.

We estimate growth of model complexities of one-hidden-layer perceptron networks representing Boolean functions with increasing numbers d of variables. We give several sufficient conditions guaranteeing for the number of units in one-hidden-layer networks linear and polynomial dependencies on d. Using the concept of Hadamard communication matrix, we describe a class of functions that do not satisfy these conditions and so might need exponentially large networks for their representations. We show that the function "inner product mod 2" belongs to this class and prove that for its representation by a network with two hidden layers, merely $d/2$ Heaviside perceptrons in each layer is sufficient. Further, we propose a mathematical formalization of the observation of Bengio et al. [17, 16] that "amount of variations of a function" can cause difficulties in its representations by one-hidden-layer networks. We use the concept of variational norm with respect to a dictionary of computational units as a measure of tractability of a representation of a function by a neural network reflecting both number of hidden units and sizes of output weights. Bartlett [18] demonstrated that in some cases, the size of weights is a more important factor for successful learning than the number of network units. We prove that in a one-hidden-layer

perceptron network representing a d-variable Boolean function with a Hadamard communication matrix, the number of units depends on d exponentially or the absolute values of some output weights increase exponentially with d.

The paper is organized as follows. Section 2 contains some concepts and notations from the area of multilayer networks, Boolean functions, and their Fourier transforms. Section 3 investigates model complexities of one and two-hidden-layer perceptron networks representing d-variable Boolean functions. Section 4 proposes a mathematical formalization of the concept of a "highly-varying function" in terms of a variational norm with respect to a type of network units. Section 5 is a discussion.

2 Preliminaries

A widely-used network architecture is a *one-hidden-layer network with a single linear output*. Such a network with n hidden units can compute input-output functions from the set

$$\operatorname{span}_n G := \left\{ \sum_{i=1}^{n} w_i g_i \,\middle|\, w_i \in \mathbb{R},\, g_i \in G \right\},$$

where G, called *dictionary*, is a set of functions computable by a given type of units. In this paper we use the term one-hidden-layer network meaning a network with a single linear output.

We investigate model complexities of networks computing functions from the space

$$\mathcal{B}(\{0,1\}^d) := \{f \,|\, f : \{0,1\}^d \to \mathbb{R}\}$$

of *real-valued Boolean functions of d variables*. As $\mathcal{B}(\{0,1\}^d)$ is isomorphic to the Euclidean space $\mathbb{R}^{2^d}$, on $\mathcal{B}(\{0,1\}^d)$ we have the Euclidean inner product defined as $\langle f, g\rangle := \sum_{u\in\{0,1\}^d} f(u)g(u)$ and the Euclidean norm $\|f\|_2 := \sqrt{\langle f, f\rangle}$. By $\cdot$ we denote the inner product on $\{0,1\}^d$ defined as $u \cdot v := \sum_{i=1}^{d} u_i v_i$.

An important subset of $\mathcal{B}(\{0,1\}^d)$ is formed by *generalized parities*. For a set $I \subseteq \{1,\ldots,d\}$, *I-parity* $p_I : \{0,1\}^d \to \{0,1\}$ is defined as $p_I(x) := 1$ if $\sum_{i\in I} x_i$ is odd and $p_I(x) := 0$ otherwise. Note that in some literature parities are considered as functions with values in $\{-1,1\}$ defined as $p_u(x) := (-1)^{x\cdot u}$, where $u \in \{0,1\}^d$. Obviously, $p_u(x) = s(p_{I_u}(x))$, where $s : \{0,1\} \to \{-1,1\}$ is defined as $s(0) := 1$ and $s(1) := -1$ and $I_u := \{i \in \{0,\ldots,d\} \,|\, u_i = 1\}$.

Let the subset $F_d := \{\phi_u \,|\, u \in \{0,1\}^d\}$ of $\mathcal{B}(\{0,1\}^d)$ be defined for every $u, x \in \{0,1\}^d$ as

$$\phi_u(x) := 2^{-d/2}\,(-1)^{x\cdot u}\,. \tag{1}$$

It is well-known and easy to check that F_d forms an orthonormal basis of $\mathcal{B}(\{0,1\}^d)$ called *Fourier basis*.

We consider two dictionaries of computational units, which are subsets of $\mathcal{B}(\{0,1\}^d)$. The first one is the set H_d of functions on $\{0,1\}^d$ computable by *Heaviside perceptrons*, i.e.,

$$H_d := \{\vartheta(e\cdot\,. + b) : \{0,1\}^d \to \{0,1\} \,|\, e \in \mathbb{R}^d, b \in \mathbb{R}\}\,, \tag{2}$$

where ϑ denotes the *Heaviside activation function* defined as $\vartheta(t) := 0$ for $t < 0$ and $\vartheta(t) := 1$ for $t \geq 0$. Note that H_d is the set of characteristic functions of half-spaces. The set H_d is much smaller than the whole space $\mathcal{B}(\{0,1\}^d)$ as it has cardinality smaller than 2^{d^2} [19]. The second dictionary that we consider, denoted by S_d, is closely related to H_d. It is formed by functions on $\{0,1\}^d$ computable by perceptrons with the *signum activation function* $\operatorname{sgn} : \mathbb{R} \to \{-1,1\}$ defined as $\operatorname{sgn}(t) := -1$ for $t < 0$ and $\operatorname{sign}(t) := 1$ for $t \geq 0$. So

$$S_d := \{\operatorname{sgn}(v \cdot . + b) : \{0,1\}^d \to \{-1,1\} \mid v \in \mathbb{R}^d, b \in \mathbb{R}\}. \tag{3}$$

3 Representations of Boolean Functions by One and Two-Hidden-Layer Perceptron Networks

In this section, we investigate model complexities of one and two-hidden-layer networks with Heaviside perceptrons representing Boolean functions.

Ito [8, Theorem 4] proved that every function on a finite subset X of $\mathbb{R}^d$ can be represented by a network with one hidden layer with sigmoidal perceptrons, the number of which does not exceed the cardinality of the set X. So in particular, every element of $\mathcal{B}(\{0,1\}^d)$ can be represented by one-hidden-layer Heaviside network with at most 2^d units. For the special case of Boolean functions with Boolean values, Ito [8, Theorem 5] proved that they can be represented by networks with the number of hidden units equal to the cardinality of their support and all output weights equal to 1. For $f : \{0,1\}^d \to \mathbb{R}$ we denote $\operatorname{supp}(f) := \{u \in \{0,1\}^d \mid f(u) \neq 0\}$.

Theorem 1 (Ito). *Let d be a positive integer, $f : \{0,1\}^d \to \{0,1\}$, and $w \in \{0,1\}^d$. Then for all $x \in \{0,1\}^d$*

$$f(x) = \sum_{u \in \operatorname{supp}(f)} \vartheta(v_u \cdot x - b_u),$$

where $v_u := 2u - w$ and b_u satisfies $\|u\|^2 - 1 < b_u \leq \|u\|^2$.

Theorem 1 implies a condition on a d-variable function $f : \{0,1\}^d \to \{0,1\}$ sufficient to be representable by a one-hidden-layer network with the number of Heaviside perceptrons being a polynomial in d.

Corollary 1. *Let d be a positive integer and $f : \{0,1\}^d \to \{0,1\}$ a function such that* $\operatorname{card}\operatorname{supp}(f)$ *or* $\operatorname{card}(\{0,1\}^d \setminus \operatorname{supp}(f))$ *is a polynomial in d. Then f can be represented by a network with one linear output and one hidden layer with the number of Heaviside perceptrons being a polynomial in d.*

Thus functions $f : \{0,1\}^d \to \{0,1\}$ whose representations by one-hidden-layer networks are not "tractable" in the sense that they require exponentially many Heaviside perceptrons, have to be searched among functions with both supports and their complements of sizes exponential in d. Obviously, this condition is not necessary. For example, the projection $\pi_1 : \{0,1\}^d \to \{0,1\}$ defined as

$\pi_1(x_1, \ldots, x_d) = x_1$ has both support and its complement of cardinality $2^{d/2}$, but it can be computed by a single perceptron $\vartheta((1, 0, \ldots, 0) \cdot (x_1, \ldots, x_d) - 1/2)$.

Generalized parities do not belong to such class of non tractable functions as they can be represented by one-hidden-layer networks having merely d hidden units. Indeed, it is easy to verify that for every $u \in \{0,1\}^d$, the function ϕ_u from the Fourier basis can be represented as $\phi_u(x) = 2^{-d/2}\,(-1)^{x \cdot u} = 2^{-d/2} \sum_{i=1}^{d} (-1)^i\, \vartheta(u \cdot x - i + 1/2)$. Thus any function with the number of non zero coefficients in its Fourier representation depending on d polynomially can be represented by a network with a polynomial number of Heaviside perceptrons. Generalized parities are symmetric functions, i.e., they are invariant under permutations of entries of vectors $x \in \{0,1\}^d$. More precisely, a function $f : \{0,1\}^d \to \mathbb{R}$ is called *symmetric* if there exists a function $g : \{0, \ldots, d\} \to \mathbb{R}$ such that for all $x \in \{0,1\}^d$, $f(x) = g(\sum_{i=1}^{d} x_i)$. The following proposition shows that any symmetric function of d variables can be represented by a network with d Heaviside perceptrons.

Proposition 1. *Let d be a positive integer. Then every symmetric function $f : \{0,1\}^d \to \mathbb{R}$ can be represented by a one-hidden-layer network with d Heaviside perceptrons.*

Proof. Let $f(x) = g(\sum_{i=1}^{d} x_i)$. Then $f(x) = g(0)\, \vartheta(x \cdot (1, \ldots, 1) + 1/2) + \sum_{i=1}^{d-1} (g(i) - g(i-1))\, \vartheta(x \cdot (1, \ldots, 1) - i + 1/2)$. □

Thus searching for functions which might not be representable by one-hidden-layer networks with polynomially many perceptrons one has to look for more complicated functions than generalized parities and symmetric functions.

By $*$ we denote the *concatenation* of two vectors in $\{0,1\}^k$, i.e., for $u, v \in \{0,1\}^k$, $u * v \in \{0,1\}^{2k}$ such that

$$(u * v)_i = u_i \text{ for } i = 1, \ldots, k \text{ and } (u * v)_i = v_i \text{ for } i = k+1, \ldots, 2k.$$

A *communication matrix of a function* $f : \{0,1\}^d \to \{-1,1\}$ is a $2^{d/2} \times 2^{d/2}$ matrix $M(f)$ with rows and columns indexed by vectors $u, v \in \{0,1\}^{d/2}$, where

$$M(f)_{u,v} := f(u * v).$$

A *Hadamard matrix* is a square matrix M with entries in $\{-1,1\}$ such that any two distinct columns (or equivalently rows) of M are orthogonal. For d even, let $\beta_d : \{0,1\}^d \to \{0,1\}$ denote the function *inner product mod* 2, defined for all $x \in \{0,1\}^d$ as

$$\beta_d(x) := 1 \text{ if } l(x) \cdot r(x) \text{ is odd and } \beta_d(x) := 0 \text{ if } l(x) \cdot r(x) \text{ is even,}$$

where $l(x), r(x) \in \{0,1\}^{d/2}$ are set for every $i = 1, \ldots d/2$ as $l(x)_i := x_i$ and $r(x)_i := x_{d/2+i}$. For technical reasons, sometimes we use instead of the inner product mod 2 the function $\bar{\beta}_d : \{0,1\}^d \to \{-1,1\}$ defined as

$$\bar{\beta}_d(x) := (-1)^{l(x) \cdot r(x)}. \tag{4}$$

To show that the function inner product mod 2 has both the preimages of 1 and of 0 of sizes exponential in d, we use a lemma by Lindsay (see, e.g., [20, p.88]), which estimates the difference between the numbers of 1s and -1s in submatrices of Hadamard matrices.

Lemma 1 (Lindsey) *Let n be a positive integer and let M an $n \times n$ Hadamard matrix. Let A, B be subsets of the set of indices of rows, columns, resp., of M. Then $\left|\sum_{a\in A}\sum_{b\in B} M_{a,b}\right| \leq \sqrt{n \operatorname{card} A \operatorname{card} B}$.*

Proposition 2. *Let d be an even integer. Then both $\beta_d^{-1}(\{1\})$ and $\beta_d^{-1}(\{0\})$ have cardinalities exponential in d.*

Proof. It is easy to show that the communication matrix $M(\bar{\beta}_d)$ is a Hadamard matrix. Indeed, its rows multiplied by $2^{-d/2}$ form the Fourier basis $F_{d/2}$ of $\mathcal{B}(\{0,1\}^{d/2})$. Let $\alpha := \operatorname{card}(\bar{\beta}_d^{-1}(\{1\})$ and $\beta := \operatorname{card}(\bar{\beta}_d^{-1}(\{-1\})$. So by the definition, $\alpha+\beta = 2^d$. As the 2^d entries of $2^{d/2} \times 2^{d/2}$ matrix $M(\bar{\beta}_d)$ represent values of $\bar{\beta}_d(u)$ for all $u \in \{0,1\}^d$, by Lemma 1 we have $|\alpha - \beta| \leq 2^{3d/4}$. On the other hand, $\alpha+\beta = 2\min(\alpha,\beta)+|\alpha-\beta| = 2^d$. Hence, $|\alpha-\beta| = 2^d-2\min(\alpha,\beta) \leq 2^{3d/4}$. Thus $2^d - 2^{3d/4} \leq 2\min(\alpha,\beta)$, hence α and β must be exponential in d. □

Proposition 2 implies that applying the representation from Theorem 1 to β_d provides a network with the number of perceptrons exponential in d.

On the other hand, the next theorem shows that when Heaviside perceptrons are arranged in two hidden layers, the function $\bar{\beta}_d$ can be computed by a network with only d hidden units. Its proof exploits a representation of $\bar{\beta}_d$ as a composition of two functions, each representable by a one-hidden-layer network with a number of units dependent on d linearly.

Theorem 2. *For every even integer d, the function $\bar{\beta}_d : \{0,1\}^d \to \{-1,1\}$ can be represented by a network with one linear output and two hidden layers with $d/2$ Heaviside perceptrons each.*

Proof. For any $b \in (1,2)$, define $d/2$ perceptrons with d inputs in the first hidden layer as $\vartheta(v^i \cdot x - b)$, where $v_i^i := 1$, $v_{d/2+i}^i := 1$, and all other weights are equal to 0. So for an input vector $x \in \{0,1\}^d$, the output $y_i(x)$ of the i-th perceptron in the first hidden layer satisfies $y_i(x) = \vartheta(v^i \cdot x - b) = 1$ if and only if both $x_i = 1$ and $x_{d/2+i} = 1$, otherwise $y_i(x)$ is equal to zero.

Let $w = (w_1, \ldots, _{d/2}) = (1, \ldots, 1)$ be such that $w_j := 1$ for all $j = 1, \ldots, d/2$. In the second hidden layer, define $d/2$ perceptrons $z_j(y) = \vartheta(w \cdot y - j + 1/2)$. Finally, for all $j = 1, \ldots, d/2$ let the j-th unit from the second hidden layer be connected with one linear output unit with the weight $(-1)^j$.

Thus the two-hidden-layer network computes the function $\sum_{j=1}^{d/2}(-1)^j \vartheta(w \cdot y(x) - j + 1/2)$ where $y_i(x) = \vartheta(v^i \cdot x - b)$, i.e., it computes the function $\sum_{j=1}^{d/2}(-1)^j \vartheta(\sum_{i=1}^{d/2} \vartheta(v^i \cdot x - b) - j + 1/2)$. □

4 Highly-Varying Functions and Variation with Respect to Half-Spaces

In this section, we propose a mathematical formalization of the concept of "highly varying functions" suggested by Bengio et al. [16, 17] as "difficult" functions for computation by one-hidden-layer networks. We suggest that the concept of a variational norm tailored to a dictionary of computational units, which plays an important role in estimates of rates of approximation, can serve as a measure of "tractability" of a representation of a function by a network with units from a finite dictionary.

Variation with respect to a set of functions was introduced by Kůrková [21] as an extension of Barron's [22] concept of variation with respect to characteristic functions. Barron considered the set of characteristic functions of halfspaces corresponding to the dictionary of functions computable by Heaviside perceptrons. Variational norms play an important role in estimates of rates of approximation by one-hidden-layer networks (see, e.g., [10] and the references therein).

For a subset G of a normed linear space $(\mathcal{X}, \|.\|_{\mathcal{X}})$, *$G$-variation (variation with respect to the set G)*, denoted by $\|.\|_G$, is defined as

$$\|f\|_G := \inf \{c \in \mathbb{R}_+ \mid f/c \in \mathrm{cl}_{\mathcal{X}} \operatorname{conv} (G \cup -G)\} , \tag{5}$$

where $\mathrm{cl}_{\mathcal{X}}$ denotes the closure with respect to the norm $\|\cdot\|_{\mathcal{X}}$ on $\mathcal{X}$. It was shown in [23] that the infimum in the definition (5) can be replaced by minimum.

For a finite dictionary G, G-variation of a function $f \in \operatorname{span} G$ is equal to the minimal sum of absolute values of coefficients in all possible representations of f as linear combinations of elements of G. More precisely, by [24, Proposition 2.3] for G with $\operatorname{card} G = m$ and $f \in \operatorname{span} G$ we have

$$\|f\|_G = \min \left\{ \sum_{i=1}^{m} |w_i| \,\Big|\, f = \sum_{i=1}^{m} w_i \, g_i \,,\; w_i \in \mathbb{R},\; g_i \in G \right\} . \tag{6}$$

So if a function has a "large" variation with respect to a finite set G, then each its representation by a network with units from G has either "large number of units" or some units have "large" output weights.

We investigate variation with respect to the two dictionaries, H_d and S_d, defined in Section 2, formed by Boolean functions computable by Heaviside perceptrons and signum perceptrons, resp.

The following theorem shows that variations with respect to half-spaces of functions with Hadamard communication matrices grow exponentially with d. The proof follows from [24, Theorem 3.7] and the relationship between S_d-variation and H_d-variation. Following [25], we use the notation $h = \Omega(g(d))$ for two functions $g, h : \mathbb{N} \to \mathbb{R}$ if there exist a positive constant c and $n_0 \in \mathbb{N}$ such that for all $n \geq n_o$, $h(n) \geq c\, g(n)$.

Theorem 3. *Let d be an even integer and $f : \{0,1\}^d \to \{-1,1\}$ a function with a Hadamard communication matrix. Then $\|f\|_{H_d} = \Omega(2^{2d/3})$.*

Proof. It was shown in [24, Theorem 3.7] that under the assumptions of the theorem, $\|f\|_{S_d} = \Omega(2^{d/6})$. Thus it is sufficient to verify that for every $f \in \mathcal{B}(\{0,1\}^d)$, $\|f\|_{S_d} \leq \|f\|_{H_d}$. By [26, Proposition 3], for every pair of subsets F, G of a normed linear space, $\|.\|_G \leq c\|.\|_F$ iff for all $h \in F$, $\|h\|_G \leq c$. Since $\vartheta(e \cdot x + b) = \frac{1}{2}\mathrm{sgn}(e \cdot x + b) + \frac{1}{2} = \frac{1}{2}\mathrm{sgn}(e \cdot x + b) + \frac{1}{2}\mathrm{sgn}(e \cdot (1, \ldots, 1) + 1)$, we have for every $h \in H_d$, $\|h\|_{S_d} \leq 1$. So $\|.\|_{S_d} \leq \|.\|_{H_d}$ and the statement holds. □

By Theorem 3 and the formula (6) we get the next corollary.

Corollary 2. *Let d be an even integer, $f : \{0,1\}^d \to \{-1,1\}$ a function with a Hadamard communication matrix and $f(x) = \sum_{i=1}^m w_i \vartheta(e_i \cdot x + b_i)$ be its representation by a one-hidden-layer Heaviside perceptron network. Then $\sum_{i=1}^m |w_i| = \Omega(2^{2d/3})$.*

By Corollary 2, if a d-variable Boolean function with a Hadamard communication matrix can be represented by a one-hidden-layer Heaviside perceptron network with the number of units depending on d merely polynomially, then some of the network output weights must have exponentially large sizes. In the proof of Proposition 2, it was verified that $\bar{\beta}_d$ has a Hadamard communication matrix. So we get the next Corollary.

Corollary 3. *Let d be an even integer and $\bar{\beta}_d(x) = \sum_{i=1}^m w_i \vartheta(e_i \cdot x + b_i)$ a representation of the function $\bar{\beta}_d$. Then $\sum_{i=1}^m |w_i| = \Omega(2^{2d/3})$.*

So in a representation of the d-dimensional function inner product mod 2 by a one-hidden-layer Heaviside perceptron network, the number of units must be exponential in d or some output weights must have absolute values of sizes exponential in d. On the other hand, Theorem 2 shows that there exists a representation of this function by a two-hidden-layer network with merely $d/2$ units in each hidden layer.

5 Discussion

We addressed the difficulty of efficiently representing d-variable Boolean functions by networks with merely one hidden layer and advantages of using two hidden layers. To get some insight into properties that make some Boolean functions hardly representable by networks with only one hidden layer, we estimated numbers of units in representing networks in dependence on their input dimensions d.

We derived conditions for Boolean functions guaranteeing representations by networks with the numbers of perceptrons depending on d polynomially. We described Boolean functions that do not satisfy these conditions. We proposed a formalization of the concept of "amount of variations of a function" suggested by Bengio et al. [16, 17] as an important factor for tractability of its representation by a neural network. We showed that when a Boolean function has a "large" variational norm with respect to a dictionary of computational units, then each

its representation by a network with units from the dictionary has "large" number of units or "large" size of output weights. In particular for functions with Hadamard communication matrices, the number of Heaviside perceptrons in one-hidden-layer networks representing these functions depends on d exponentially or the absolute values of some output weights must increase exponentially with d. As an archetype of functions with such a behavior we presented the function inner product mod 2 and used it to demonstrate an example of a function which can be represented more efficiently by a network with two hidden layers than a network with merely one hidden layer.

The question whether a one-hidden-layer Heaviside perceptron network with one linear output can represent the inner product mod 2 with a number of units that grows with d polynomially is an open problem. Hajnal et al. [27, 28] derived an exponential lower bound on the number of hidden units needed to compute the inner product mod 2 by a one-hidden-layer Heaviside perceptron network with a single output Heaviside perceptron and all the weights between hidden units and the output unit being integers bounded by a polynomial in d.

Acknowledgments. V.K. was partially supported by GA ČR grant P202/11/1368 and institutional support 67985807. The authors thank P. C. Kainen for fruitful discussions.

References

1. Fine, T.L.: Feedforward Neural Network Methodology. Springer, Heidelberg (1999)
2. Chow, T.W.S., Cho, S.Y.: Neural Networks and Computing: Learning Algorithms and Applications. World Scientific (2007)
3. Leshno, M., Lin, V.Y., Pinkus, A., Schocken, S.: Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. Neural Networks 6, 861–867 (1993)
4. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numerica 8, 143–195 (1999)
5. Park, J., Sandberg, I.: Approximation and radial-basis-function networks. Neural Computation 5, 305–316 (1993)
6. Mhaskar, H.N.: Versatile Gaussian networks. In: Proceedings of IEEE Workshop of Nonlinear Image Processing, pp. 70–73 (1995)
7. Kůrková, V.: Some Comparisons of Networks with Radial and Kernel Units. In: Villa, A.E.P., Duch, W., Érdi, P., Masulli, F., Palm, G. (eds.) ICANN 2012, Part II. LNCS, vol. 7553, pp. 17–24. Springer, Heidelberg (2012)
8. Ito, Y.: Finite mapping by neural networks and truth functions. Mathematical Scientist 17, 69–77 (1992)
9. Micchelli, C.A.: Interpolation of scattered data: Distance matrices and conditionally positive definite functions. Constructive Approximation 2, 11–22 (1986)
10. Kainen, P.C., Kůrková, V., Sanguineti, M.: Dependence of computational models on input dimension: Tractability of approximation and optimization tasks. IEEE Trans. on Information Theory 58(2), 1203–1214 (2012)
11. Maiorov, V., Pinkus, A.: Lower bounds for approximation by MLP neural networks. Neurocomputing 25, 81–91 (1999)

12. Maiorov, V.: On best approximation by ridge functions. J. of Approximation Theory 99, 68–94 (1999)
13. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural Computation (2006)
14. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning 2, 1–127 (2009)
15. Grochowski, M., Duch, W.: Learning Highly Non-separable Boolean Functions Using Constructive Feedforward Neural Network. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) ICANN 2007, Part I. LNCS, vol. 4668, pp. 180–189. Springer, Heidelberg (2007)
16. Bengio, Y., Delalleau, O., Roux, N.L.: The curse of highly variable functions for local kernel machines. In: Advances in Neural Information Processing Systems, vol. 18, pp. 107–114. MIT Press (2006)
17. Bengio, Y., Delalleau, O., Roux, N.L.: The curse of dimensionality for local kernel machines. Technical Report 1258, Département d'Informatique et Recherche Opérationnelle, Université de Montréal (2005)
18. Bartlett, P.L.: The sample complexity of pattern classification with neural networks: The size of the weights is more important than the size of the network. IEEE Trans. on Information Theory 44, 525–536 (1998)
19. Shläfli, L.: Gesamelte mathematische abhandlungen, Band 1 (1950)
20. Erdös, P., Spencer, J.H.: Probabilistic Methods in Combinatorics. Academic Press (1974)
21. Kůrková, V.: Dimension-independent rates of approximation by neural networks. In: Warwick, K., Kárný, M. (eds.) Computer-Intensive Methods in Control and Signal Processing, pp. 261–270. Birkhäuser, Boston (1997)
22. Barron, A.R.: Neural net approximation. In: Narendra, K. (ed.) Proc. 7th Yale Workshop on Adaptive and Learning Systems, pp. 69–72. Yale University Press (1992)
23. Kůrková, V.: Complexity estimates based on integral transforms induced by computational units. Neural Networks 33, 160–167 (2012)
24. Kůrková, V., Savický, P., Hlaváčková, K.: Representations and rates of approximation of real–valued Boolean functions by neural networks. Neural Networks 11, 651–659 (1998)
25. Knuth, D.E.: Big omicron and big omega and big theta. SIGACT News 8(2), 18–24 (1976)
26. Kůrková, V., Sanguineti, M.: Comparison of worst-case errors in linear and neural network approximation. IEEE Trans. on Information Theory 48, 264–275 (2002)
27. Hajnal, A., Maas, W., Pudlák, P., Szegedy, M., Turán, G.: Threshold circuits of bounded depth. In: Proc. 28th Annual Symposium on Foundations of Computer Science, pp. 99–110. IEEE (1987)
28. Hajnal, A., Maas, W., Pudlák, P., Szegedy, M., Turán, G.: Threshold circuits of bounded depth. J. of Computer and System Sciences 46, 129–154

Time Series Visualization Using Asymmetric Self-Organizing Map

Dominik Olszewski[1], Janusz Kacprzyk[2], and Sławomir Zadrożny[2]

[1] Faculty of Electrical Engineering,
Warsaw University of Technology, Poland
dominik.olszewski@ee.pw.edu.pl
[2] Systems Research Institute,
Polish Academy of Sciences, Poland
{janusz.kacprzyk,slawomir.zadrozny}@ibspan.waw.pl

Abstract. We propose an asymmetric version of the Self-Organizing Map (SOM) capable to properly visualize datasets consisting of time series. The goal is achieved by introducing an asymmetric coefficient making the asymmetric SOM capable to handle time series. The experiments on the U.S. Stock Market Dataset verify and confirm the effectiveness of the proposed asymmetric SOM extension.

Keywords: Self-Organizing Map, asymmetric Self-Organizing Map, asymmetry, visualization, U.S. Stock Market visualization.

1 Introduction

The Self-Organizing Map (SOM) by T. Kohonen [1] is an example of the artificial neural network architecture. The approach can be also interpreted as a visualization technique, since the algorithm performs a projection from multidimensional space to 2-dimensional space, this way creating a map structure. The location of points in 2-dimensional grid aims to reflect the similarities between the corresponding objects in multidimensional space. Therefore, the SOM algorithm allows for visualization of relationships between objects in multidimensional space.

The asymmetric version of the SOM algorithm (i.e., SOM using the asymmetric similarities) was introduced in [2], and it was extended in [3].

However, the methodology utilized in those papers does not apply to every kind of analyzed data. In particular, at it was discussed in [3], time series are an example of the data difficult to handle by the asymmetric SOM. This happens because, the asymmetric nature of data in [2] is derived from the hierarchical data relationships (see Section 3). The hierarchical relationships, in turn, are reflected using the asymmetric coefficients measuring the frequencies of occurrences of objects in an analyzed dataset. And, in case of the time series analysis, when each visualized object is a relatively long vector of samples, it is pointless to measure frequencies of occurrences of objects, because it is almost impossible to find two identical vectors of samples, and consequently, the method from [2]

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 40–49, 2013.

will fail. Nevertheless, the hierarchy-caused asymmetry phenomenon still exist is such case. However, this time, it should be differently observed, interpreted, and handled.

The specific character of the time series data often enforces formulating special forms of methods and algorithms designed for that particular kind of data, e.g., in case of clustering, see [4], and in case of classification, see [5].

In this paper, we propose a novel asymmetric coefficient designed for measuring the hierarchy-caused degree of asymmetry in time series datasets. The coefficient will then be used in order to weight the standard Euclidean distance, and subsequently, in order to obtain an asymmetric similarity utilized in the asymmetric SOM. The proposed coefficient is formulated this way that it finds hierarchical associations in time series dataset, even if there are no identical series.

The results of the experimental study carried out on the U.S. Stock Market Dataset verify and confirm the effectiveness of the proposed approach.

2 Symmetric Self-Organizing Map

The SOM algorithm provides a non-linear mapping between a high-dimensional original data space and a 2-dimensional map of neurons. The neurons are arranged according to a regular grid, in such a way that the similar vectors in input space are represented by the neurons close in the grid. Therefore, the SOM technique visualize the data associations in the input high-dimensional space.

It was shown in [6] that the results obtained by the SOM method are equivalent to the results obtained by optimizing the following error function:

$$e\left(\mathcal{W}\right) = \sum_{r} \sum_{x_\mu \in V_r} \sum_{s} h_{rs} D\left(x_\mu,\ w_s\right) \tag{1}$$

$$\approx \sum_{r} \sum_{x_\mu \in V_r} D\left(x_\mu,\ w_r\right) \ +\ K \sum_{r} \sum_{s \neq r} h_{rs} D\left(w_r,\ w_s\right)\ , \tag{2}$$

where x_μ are the objects in high-dimensional space, w_r and w_s are the prototypes of objects on the grid, h_{rs} is a neighborhood function (e.g., the Gaussian kernel) that transforms non-linearly the neuron distances (see [1] for other choices of neighborhood functions), $D\left(\cdot,\ \cdot\right)$ is the squared Euclidean distance, and V_r is the Voronoi region corresponding to prototype w_r. The number of prototypes is sufficiently large so that $D\left(x_\mu,\ w_s\right) \approx D\left(x_\mu,\ w_r\right) \ +\ D\left(w_r,\ w_s\right)$.

According to equation (2), the SOM error function can be decomposed as the sum of the quantization error and the topological error. The first one minimizes the loss of information, when the input patterns are represented by a set of prototypes. By minimizing the second one, we assure the maximal correlation between the prototype dissimilarities and the corresponding neuron distances, this way assuring the visualization of the data relationships in the input space.

3 Handling the Asymmetry in Data Analysis

One of the first works regarding the asymmetric view on dissimilarity is [7] by Amos Tversky. He claims that a similarity or dissimilarity may have a directional character, i.e., it may have a subject and a referent. This results in asymmetric nature of the similarity or dissimilarity. His claims were validated in his numerous psychological experiments [8], and his idea was undoubtedly an inspiration for many later works concerning the asymmetric dissimilarities and the general problem of asymmetry in data analysis.

An example continuation of the Tversky's idea appears in the work of Manuel Martín-Merino and Alberto Muñoz [2], where the asymmetric version of the Self-Organizing Map was proposed. The idea of hierarchical-caused asymmetry can be also found in [9], where the asymmetric version of the k-means clustering algorithm was introduced. The author utilized a similar assertion justifying the usage of asymmetric dissimilarities. Also, in [10], where the improved version of the asymmetric k-means algorithm was proposed, the asymmetric dissimilarity was employed as preferable over the standard symmetric one.

When an analyzed dataset appears to have asymmetric properties, the symmetric measures of similarity or dissimilarity (e.g., the most popular Euclidean distance) do not grasp to this phenomenon, and for most pairs of data points, they produce small values (similarities) or big values (dissimilarities). Consequently, they do not reflect accurately the relationships between objects. The asymmetry in a dataset arises, e.g., in case, when the data associations have a hierarchical nature, i.e., when a dataset consists of general and specific entities. In case of the dissimilarity, when it is computed in the direction – from a more general entity to a more specific one – it should be greater than in the opposite direction. As an example, one can consider the text analysis field, particularly, the dissimilarity between the two words: "Mathematics" and "Bayes". The former is a more general word, and consequently, the dissimilarity from "Mathematics" to "Bayes" should be greater than in the opposite direction, because the meaning of the word "Mathematics" contains a lot more topics than just "Bayes," while the word "Bayes" directly comes under "Mathematics." The hierarchical connections in data are closely related to the asymmetry. This relation has been noticed in [11]. As stated in [2], asymmetry can be interpreted as a particular type of hierarchy.

An idea to overcome this problem is to employ the asymmetric similarities and dissimilarities. They should be applied in algorithms in such a way, so that they would properly reflect the hierarchical asymmetric relationships between objects in an analyzed dataset. Therefore, it should be guaranteed that their application is consistent with the hierarchical associations in data. This can be achieved by use of the asymmetric coefficients, inserted in the formulae of symmetric measures. This way, we can obtain the asymmetric measures on the basis of the symmetric ones. The asymmetric coefficients should assure the consistency with the hierarchy. Hence, in case of the dissimilarities, they should assure greater values in the direction – from a more general concept to a more specific one.

In this paper, we deal with the datasets consisting of time series, and our asymmetric SOM version is time-series-oriented.

Our paper proposes an asymmetric coefficient, which can be successfully used in case of the time series analysis, in contrast to the prior work [2], where the object occurrences frequencies are used in order to asymmetrize the dissimilarities, which essentially inhibits and limits the possibility of application of the approach from [2] in the field of time series visualization.

3.1 Asymmetric Coefficient

The role of the asymmetric coefficient is to convey the information provided by the asymmetry. Two coefficients were introduced in [12]. The first one is derived from the fuzzy logic similarity, and the second one formulated on the basis of the Kullback-Leibler divergence.

Time series are usually relatively long vectors of samples. Even after effective feature extraction and dimensionality reduction, they still usually remain highly multidimensional. Hence, it is difficult to find two identical instances of time series. Therefore, both coefficients mentioned in [12] are not a recommended choice in that case.

The asymmetric coefficient, proposed in this paper, is designed for the time series analysis. It measures the frequencies of occurrences of features with a given tolerance. This kind of approach makes it possible to divide the time series in an analyzed dataset to the ones more general, and to the ones more specific, even if there are no identical time series in a given dataset. Our method allows for identifying time series belonging to different levels of generality. Consequently, the hierarchical relationships can be reflected and exploited in datasets. The mentioned tolerance is set arbitrarily, and in our experimental research was set empirically (see Section 5.3).

The proposed asymmetric coefficient is formulated in the following way:

$$a_i = \frac{|f_i \pm \delta|}{\max_j (|f_j \pm \delta|)}, \tag{3}$$

where f_i are the features of time series in an analyzed dataset, $|\cdot|$ is the norm meaning the number of time series possessing the features from the interval $\langle f_i - \delta, f_i + \delta \rangle$, and δ is a chosen tolerance belonging to the interval $\langle 0, \max_j (|f_j|) \rangle$.

This coefficient takes values in the $\langle 0, 1 \rangle$ interval. Intuitively speaking, it will become large for general (broad) concepts with large $|\cdot|$ norm.

Note that the asymmetric coefficients must be computed and assigned to each feature of every time series instance in an analyzed dataset.

4 Asymmetric Self-Organizing Map

On the basis of the asymmetric coefficient introduced in Section 3, the asymmetric version of the SOM algorithm will be formulated. Since the coefficient

a_i is designed for the time series analysis, the asymmetric SOM considered in this section will also be time-series-oriented, and appropriate terminology will be used. In order to obtain the target asymmetric SOM, we will refer to the error function (2). As it was stated in Section 2, the results produced by the SOM method are identical to the results obtained by optimizing the function (2).

The asymmetric SOM algorithm is derived in three steps:

Step 1. Transform a symmetric dissimilarity (e.g., the Euclidean distance) into a similarity:

$$S_{ij}^{\mathrm{SYM}} = C - d^2(x_i, x_j) , \tag{4}$$

where $d^2(x_i, x_j)$ is the squared Euclidean distance between objects x_i and x_j, and the constant C is the upper boundary of the squared Euclidean distance over all the pairs of objects belonging to the dataset in question. This step is necessary, because the asymmetric coefficient a_i defined in (3) imposes the requirement of using a similarity instead of a dissimilarity (a_i measures certain frequency).

Step 2. Transform the symmetric similarity into the asymmetric similarity:

$$S_{ij}^{\mathrm{ASYM}} = a_i \left(C - d^2(x_i, x_j)\right) , \tag{5}$$

where a_i is the asymmetric coefficient defined in Section 3, in (3), and the rest of notation is described in (4). The asymmetric similarity defined this way, using the proposed asymmetric coefficient guarantees the consistency with the asymmetric hierarchical associations among the time series in the dataset (explained in Section 3).

Step 3. Insert the asymmetric similarity in the error function (2), in order to obtain the energy function, which needs to be maximized:

$$E(\mathcal{W}) = \sum_r \sum_{x_\mu \in V_r} \sum_s h_{rs} a_i \left(C - d^2(x_i, x_j)\right) , \tag{6}$$

where the notation is explained in (2), (4), and (5). The energy function (6) can be optimized in the similar way as the error function (2). The update formula of the asymmetric SOM is similar to the adaptation rule of the standard symmetric SOM, with the difference that the asymmetric coefficient is inserted. Employing the similarity (derived in Steps 1 and 2) instead of dissimilarity results in changing the SOM objective function from the error function (2) to the energy function (6). Consequently, the SOM optimization process changes from minimization to maximization. An important property of the asymmetric SOM is that it maintains the computational simplicity of the symmetric approach.

5 Experiments

In our experimental study, we have compared the proposed asymmetric SOM and the traditional symmetric SOM, both visualizing the time series.

The visualizations provided by the two SOM versions have been used as an input to the traditional k-means clustering algorithm, i.e., the clustering was performed in the 2-dimensional space, and the clustering results have been used as the basis of the comparison between the two investigated techniques. The clustering results have been assessed using two evaluation criteria, i.e., the accuracy degree and the entropy measure, described in Section 5.2.

5.1 Dataset Description

The experiments have been carried out on the U.S. Stock Market Dataset. The dataset consisted of time series representing the close prices of five stocks from the U.S. Stock Market adjusted for dividends and splits. The adjusted close prices have been collected in the time interval from 27.11.1992 to 28.09.2012. As a result, each of the analyzed time series consisted of 5000 samples. The following stocks have been considered: Bank of America Corporation, Boeing Co., General Electric Co., Intel Corporation, Time Warner Inc. Each of the time series corresponding to the respective stock has been divided into 10 subseries. Consequently, we have obtained the dataset consisting of 50 time series, each consisting of 500 samples. In a feature extraction process, 25 features have been extracted from the time series, hence, the dimensionality of the analyzed data has been reduced to 25 dimensions. Then, the 50 time series have been visualized using symmetric and asymmetric SOM, and subsequently, clustered using the traditional k-means clustering algorithm. The clustering process aimed to separate time sub-series corresponding to the same stock.

The feature extraction has been conducted using the standard Discrete-Fourier-Transform-based method.

In the first part of our experiments, we have visualized, and then clustered only 3 stocks from the U.S. Stock Market Dataset, i.e., Boeing Co., Intel Corporation, and Time Warner Inc., while in the second part of our experimental study, we have taken into account all 5 stocks from the U.S. Stock Market Dataset.

5.2 Evaluation Criteria

In our empirical research, we have compared the results of the k-means clustering of the symmetric and asymmetric SOM visualizing the time series. As the basis of the comparisons, i.e., as the evaluation criteria, we have used the accuracy degree [9,3], and the entropy measure [2,3].

Hence, the following two evaluation criteria have been used:

1. **Accuracy degree.** This evaluation criterion determines the number of correctly assigned objects divided by the total number of objects.

   Hence, for the ith formed cluster, the accuracy degree is determined as follows:

$$q_i = \frac{m_i}{n_i}, \tag{7}$$

where m_i, $i = 1, \ldots, k$ is the number of objects correctly assigned to the ith cluster, n_i, $i = 1, \ldots, k$ is the number of objects in the ith cluster, and k is the number of clusters.

And, for the entire dataset, the total accuracy degree is determined as follows:

$$q_{\text{total}} = \frac{m}{n}, \tag{8}$$

where m is the total number of correctly assigned objects, and n is the total number of objects in the entire dataset.

The accuracy degrees q_i and the total accuracy degree q_{total} assume values in the interval $\langle 0, 1 \rangle$, and naturally, greater values are preferred.

The total accuracy degree q_{total} was used in our experimental study as the main basis of the clustering accuracy comparison of the two investigated SOM approaches – the symmetric and the asymmetric.

2. **Entropy measure.** This evaluation criterion determines the number of overlapping objects divided by the total number of objects in a dataset. This means, the number of objects, which are in the overlapping area between clusters, divided by the total number of objects. If the ratio of similarities between a given object and the two nearest cluster centroids is in the interval $\langle 0.9, 1.1 \rangle$, then the object is said to be in the overlapping area. In other words, the entropy measure determines the clustering uncertainty.

   The entropy measure is determined as follows:

$$I = \frac{\mu}{n}, \tag{9}$$

   where μ is the number of overlapping objects in a dataset, and n is the total number of objects in the dataset.

   The entropy measure assumes values in the interval $\langle 0, 1 \rangle$, and, smaller values are desired.

5.3 Parameter δ

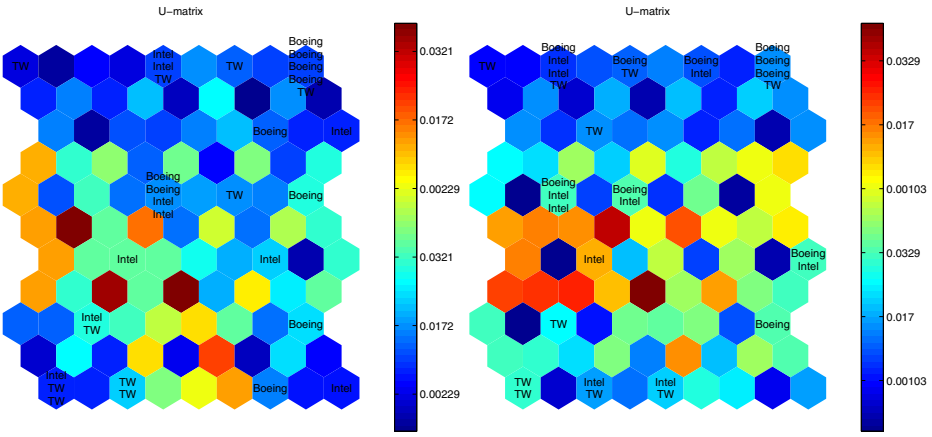
As it was stated in Section 3.1, the tolerance in comparing the features retrieved from time series is expressed using the parameter δ. During the experimental study, it was set empirically to the value of 0.005, which corresponded to the highest clustering performance among the other empirically tested values of the parameter δ.

5.4 Experimental Results

The results of our experiments are shown in Figs. 1 and 2, and in Tables 1 and 2. Figure 1 presents the U-matrices generated by the symmetric (Fig. 1(a)) and asymmetric (Fig. 1(b)) SOM techniques visualizing 3 stocks from the U.S. Stock Market Dataset, while Fig. 2 demonstrates the U-matrices generated by the symmetric (Fig. 2(a)) and asymmetric (Fig. 2(b)) SOM techniques visualizing 5 stocks from the U.S. Stock Market Dataset. The U-matrix is a graphical

presentation of SOM. Each entry of the U-matrix corresponds to a neuron on the SOM grid, while value of that entry is the average dissimilarity between the neuron and its neighbors. Table 1 reports the accuracy degrees and the entropy measures corresponding to the symmetric and asymmetric SOM methods for 3 formed clusters, while Table 2 shows the accuracy degrees and the entropy measures corresponding to the symmetric and asymmetric SOM methods for 5 formed clusters.

In case of the 3 formed clusters, the following stocks have been taken into account: Boeing Co., Intel Corporation, Time Warner Inc. In case of the 5 formed clusters, all 5 stocks in the U.S. Stock Market Dataset have been considered.



(a) Symmetric SOM (b) Asymmetric SOM

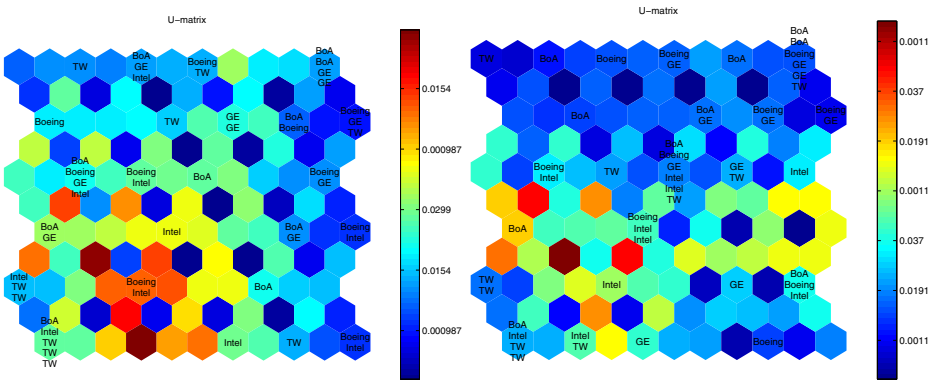
Fig. 1. Visualizations of 3 stocks from the U.S. Stock Market Dataset

In Figs. 1 and 2, the following abbreviations have been used: BoA stands for Bank of America Corporation, Boeing stands for Boeing Co., GE stands for General Electric Co., Intel stands for Intel Corporation, while TW stands for Time Warner Inc.

Table 1. Accuracy degrees and entropy measures of the 3 stocks clustering

	Symmetric SOM	Asymmetric SOM
Accuracy degree	$42/50 = 0.8400$	$45/50 = 0.9000$
Entropy measure	$9/50 = 0.1800$	$5/50 = 0.1000$

The results of both parts of our empirical research show that the proposed asymmetric SOM visualizing time series outperforms its symmetric counterpart.



(a) Symmetric SOM (b) Asymmetric SOM

Fig. 2. Visualizations of all 5 stocks from the U.S. Stock Market Dataset

Table 2. Accuracy degrees and entropy measures of the 5 stocks clustering

	Symmetric SOM	Asymmetric SOM
Accuracy degree	34/50 = 0.6800	38/50 = 0.7600
Entropy measure	18/50 = 0.3600	11/50 = 0.2200

This claim of superiority was ascertained on the basis of the clustering performance of the two investigated SOMs. By using the two evaluation criteria (accuracy degree and entropy measure), we can assert that the k-means clustering of the proposed asymmetric SOM leads to higher clustering accuracy (0.9000 vs. 0.8400 – for 3 clusters, and 0.7600 vs. 0.6800 – for 5 clusters), and also, it leads to lower clustering uncertainty (0.1000 vs. 0.1800 – for 3 clusters, and 0.2200 vs. 0.3600 – for 5 clusters). This clustering quality comparison can be used to verify and confirm the effectiveness of the proposed approach, and its superiority over the standard symmetric SOM technique.

6 Summary and Concluding Remarks

In this paper, an asymmetric coefficient a_i was introduced in Section 3. The coefficient was subsequently utilized in order to build the asymmetric SOM in Section 4. The obtained form of the asymmetric SOM is capable to effectively handle datasets consisting of time series, in contrast to the proposal of the paper [2], where the basic (not handling time series) version of the asymmetric SOM approach was presented.

The experimental research on the U.S. Stock Market Dataset showed that the proposed method outperforms the traditional symmetric one. The superiority was ascertained on the basis of the clustering performance of the two examined SOMs (symmetric and asymmetric). Both analyzed SOMs were clustered by means of the k-means clustering algorithm, and two cases (3 and 5 formed

clusters) were considered. In both these cases the symmetric SOM remained inferior with respect to the proposed technique using the introduced asymmetric coefficient a_i. Of course, further experiments are needed to more widely confirm this superiority.

Acknowledgments. The work was supported by the project "Information Technologies: Research and Their Interdisciplinary Applications" of the Human Capital Operational Programme (co-financed by the European Social Fund), which was coordinated by the Institute of Computer Science of the Polish Academy of Sciences. Also, this work was partially supported by the National Science Centre (contract no. UMO-2011/01/B/ST6/06908).

References

1. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer (2001)
2. Martín-Merino, M., Muñoz, A.: Visualizing Asymmetric Proximities with SOM and MDS Models. Neurocomputing 63, 171–192 (2005)
3. Olszewski, D.: An Experimental Study on Asymmetric Self-Organizing Map. In: Yin, H., Wang, W., Rayward-Smith, V. (eds.) IDEAL 2011. LNCS, vol. 6936, pp. 42–49. Springer, Heidelberg (2011)
4. Lee, Y.H., Wei, C.P., Cheng, T.H., Yang, C.T.: Nearest-Neighbor-Based Approach to Time-Series Classification. Decision Support Systems 53(1), 207–217 (2012)
5. D'Urso, P., Maharaj, E.A.: Wavelets-Based Clustering of Multivariate Time Series. Fuzzy Sets and Systems 193, 33–61 (2012)
6. Heskes, T.: Self-Organizing Maps, Vector Quantization, and Mixture Modeling. IEEE Transactions on Neural Networks 12(6), 1299–1305 (2001)
7. Tversky, A.: Features of Similarity. Psychological Review 84(4), 327–352 (1977)
8. Tversky, A.: Preference, Belief, and Similarity (Selected Writings). A Bradford Book, The MIT Press, Cambridge, Massachusetts (2004)
9. Olszewski, D.: Asymmetric *k*-Means Algorithm. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 1–10. Springer, Heidelberg (2011)
10. Olszewski, D.: *k*-Means Clustering of Asymmetric Data. In: Corchado, E., Snášel, V., Abraham, A., Woźniak, M., Graña, M., Cho, S.-B. (eds.) HAIS 2012, Part III. LNCS, vol. 7208, pp. 243–254. Springer, Heidelberg (2012)
11. Muñoz, A., Martin, I., Moguerza, J.M.: Support Vector Machine Classifiers for Asymmetric Proximities. In: Kaynak, O., Alpaydın, E., Oja, E., Xu, L. (eds.) ICANN 2003 and ICONIP 2003. LNCS, vol. 2714, pp. 217–224. Springer, Heidelberg (2003)
12. Muñoz, A., Martín-Merino, M.: New Asymmetric Iterative Scaling Models for the Generation of Textual Word Maps. In: Proceedings of the International Conference on Textual Data Statistical Analysis, JADT 2002, pp. 593–603 (2002)

Intelligence Approaches Based Direct Torque Control of Induction Motor

Moulay Rachid Douiri and Mohamed Cherkaoui

Mohammadia Engineering School, Department of Electrical Engineering,
Avenue Ibn Sina, B.P.765, Agdal-Rabat, Morocco
douirirachid@hotmail.com

Abstract. This paper presents a comparative study of two intelligent techniques to replace conventional comparators and selection table of direct torque control for induction machines, namely fuzzy logic and artificial neural network. The comparison with the conventional direct torque control proves that FL-DTC and NN-DTC reduces the electromagnetic torque ripple, stator flux, and stator current. Simulation results prove the effectiveness and the performances proposed strategies.

Keywords: artificial neural network, direct torque control, fuzzy logic, induction motor.

1 Introduction

A simplified variation of field orientation known as direct torque control (DTC) was developed by Takahashi [1]-[2] and Depenbrock [3]. In direct torque controlled induction motor drives, it is possible to control directly the stator flux linkage and the electromagnetic torque by the selection of an optimum inverter switching state. The selection of the switching state is made to restrict the flux and the torque errors within their respective hysteresis bands and to obtain the fastest torque response and highest efficiency at every instant [4]-[5]. DTC is simpler than field-oriented control and less dependent on the motor model, since the stator resistance value is the only machine parameter used to estimate the stator flux [6].

High torque ripple is one of the disadvantages of DTC [5]. Under constant load in steady state, an active switching state causes the torque to continue to increase past its reference value until the end of the switching period; then a zero voltage vector is applied for the next switching period causing the torque to continue to decrease below its reference value until the end of the switching period. That results in high torque ripple. A possible solution to reduce the torque ripple is to use a high switching frequency; however, that requires expensive processors and switching devices [7]-[8]. A less expensive solution is to use artificial intelligence control. In this article we propose two intelligent approaches namely fuzzy logic and artificial neural networks to replace conventional hysteresis comparators and selection table.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 50–59, 2013.

An artificial neural network (ANN) is essentially a way to learn the relationship between a set of input data and the corresponding output data. That is, it can memorize data, generalize this information when given new input data, and adjust when the relationship changes. The training is normally done with input-output examples. After training, ANNs have the capability of generalization. That is, given previously unseen input data, they can interpolate from the previous training data [9]-[10]. Inspired by the functioning of biological neurons, ANN became popular in the research community when architectures were found to enable the learning of nonlinear functions and patterns [10]-[11].

The fuzzy reasoning approach can model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analysis [12]-[13]. This approach provides an efficient way to cope with imperfect information and imprecise knowledge. It offers some kind of flexibility in decision making processes and is especially useful when a process can be controlled by a skilled human without knowledge of its underlying dynamics [12]-[14]-[15].

This paper is organized as follows: The principle of direct torque control is presented in the second part, the direct torque fuzzy control is developed in the third section, section four presents a direct torque neural control, and the fifth part is devoted to illustrate the simulation performance of this control strategy, a conclusion and reference list at the end.

Symbols:

R_s , R_r	stator and rotor resistance [Ω]
i_{sd} , i_{sq}	stator current dq axis [A]
v_{sd} , v_{sq}	stator voltage dq axis [V]
L_s , L_r	stator and rotor self inductance [H]
L_m	mutual inductance [H]
λ_{sd}, λ_{sq}	dq stator flux [Wb]
λ_{rd}, λ_{rq}	dq rotor flux [Wb]
T_e	electromagnetic torque [N.m]
E_{Te}	electromagnetic torque error [N.m]
$E_{\lambda s}$	stator flux error [Wb]
φ_s	stator flux angle [rad]
ω_r	rotor speed [rad/sec]
J	inertia moment [Kg.m^2]
p_p	pole pairs
σ	leakage coefficient
t_s	sampling period [sec]

2 Direct Torque Control

The principle of DTC is to directly select voltage vectors according to the difference between reference and actual value of electromagnetic torque and stator flux linkage.

Electromagnetic torque and stator flux errors are compared in hysteresis comparators. Depending on the comparators a voltage vector is selected from a table [16]-[17]. This can be explained by looking at the two following equations of the induction motor:

$$\begin{cases} \dfrac{d\lambda_s}{dt} = -\dfrac{R_s}{L_s}\lambda_s + \dfrac{R_s L_m}{\sigma L_s L_r}\lambda_r + v_s \\ \dfrac{d\lambda_r}{dt} = \dfrac{R_s L_m}{\sigma L_s L_r}\lambda_s + \left(jp_p\omega_r - \dfrac{R_r}{\sigma L_r} \right)\lambda_r \end{cases} \quad \text{with} \quad \sigma = 1 - \frac{L_m^2}{L_s L_r} \tag{1}$$

In the following a digital control is considered with sampling period t_s very short with respect to the motor time constants. In a generic $(k+1)t_s$ instant the stator and rotor flux space-vectors can be evaluated by means of the simplified expressions:

$$\begin{cases} \lambda_{s,k+1} = \lambda_{s,k} + \dfrac{d\lambda_{s,k}}{dt}t_s = \lambda_{s,k} + \left(-\dfrac{R_s}{L_s}\lambda_{s,k} + \dfrac{R_s L_m}{\sigma L_r L_s}\lambda_{s,k} + v_{s,k} \right)t_s \\ \lambda_{r,k+1} = \lambda_{r,k} + \dfrac{d\lambda_{r,k}}{dt}t_s = \lambda_{r,k} + \left(\dfrac{R_s L_m}{\sigma L_r L_s}\lambda_{s,k} + \left(jp_p\omega_r - \dfrac{R_r}{\sigma L_r} \right)\lambda_{r,k} \right)t_s \end{cases} \tag{2}$$

In terms of stator and rotor flux the electromagnetic torque is:

$$T_{e,k} = \frac{3}{2}p_p \frac{L_m}{\sigma L_s L_r}\mathrm{Im}\left\{\lambda_{s,k}\lambda_{r,k}\right\} \tag{3}$$

where Im [ ] represents the imaginary part of the expression in brackets.
The electromagnetic torque variation $\Delta T_{e,k}$ in each sampling interval:

$$\Delta T_{e,k} = T_{e,k+1} - T_{e,k} \tag{4}$$

Can be evaluated by introducing Eq. (2) in (3) and neglecting the terms containing the square of t_s:

$$\Delta T_{e,k} = -T_{e,k}\left(\frac{R_s}{L_s} + \frac{R_r}{L_r} \right)t_s + \frac{3}{2}p_p \frac{L_m}{\sigma L_s L_r} I_m\left\{ \left(v_{s,k}\lambda_{r,k} \right) - jp_p\omega_{r,k}\left(\lambda_{s,k}\lambda_{r,k} \right) \right\}t_s \tag{5}$$

A discrete from of stator flux in a generic sampling instant can be obtained:

$$\lambda_{s,k+1} = \lambda_{s,k} + \left(v_{s,k} - R_s i_{s,k} \right)t_s \tag{6}$$

From Eq. (9) the variations of the stator flux magnitude:

$$\Delta\lambda_{s,k} = \left|\lambda_{s,k+1}\right| - \left|\lambda_{s,k}\right| \tag{7}$$

can be easily evaluated as a function of the applied voltage. By a proper analysis of Eqs. (5) and (7), useful information can be yield about the influence both on stator flux and torque of a generic inverter voltage space vector, in correspondence of a fixed operating condition. The DTC optimum switching table is shown in Table 1.

Table 1. Switching table for conventional direct torque control

$E_{\lambda s}$	E_{Te}	n_1	n_2	n_3	n_4	n_5	n_6
1	1	V_2	V_3	V_4	V_5	V_6	V_1
	0	V_7	V_0	V_7	V_0	V_7	V_0
	-1	V_6	V_1	V_2	V_3	V_4	V_5
0	1	V_3	V_4	V_5	V_6	V_1	V_1
	0	V_0	V_7	V_0	V_7	V_0	V_7
	-1	V_5	V_6	V_1	V_2	V_3	V_4

3 Fuzzy Logic Based Direct Torque Control

The structure of the switching table can be translated in the form of vague rules. Therefore, we can replace the switching table and hysteresis comparators by a fuzzy system whose inputs are the errors on the flux and torque denoted $E_{\lambda s}$ and E_{Te} and the argument φ of the flux. The output being the command signals of the voltage inverter n. The fuzziness character of this system allows flexibility in the choice of fuzzy sets of inputs and the capacity to introduce knowledge of the human expert.

The i^{th} rule R_i can be expressed as:

$$R_i\text{: if } E_{Te} \text{ is } A_i, E_{\lambda s} \text{ is } B_i, \text{ and } \varphi \text{ is } E_i, \text{ then } n \text{ is } N_i \tag{8}$$

where A_i, B_i and C_i denote the fuzzy subsets and N_i is a fuzzy singleton set.

The synthesized voltage vector n denoted by its three components is the output of the controller.

The inference method used in this paper is Mamdani's [18] procedure based on min-max decision [19]. The firing strength η_i, for i^{th} rule is given by:

$$\eta_i = \min\left(\mu_{A_i}(E_{T_e}), \mu_{B_i}(E_{\lambda_s}), \mu_{C_i}(\varphi)\right) \tag{9}$$

By fuzzy reasoning, Mamdani's minimum procedure gives:

$$\mu'_{N_i}(n) = \min\left(\eta_i, \mu_{N_i}(n)\right) \tag{10}$$

where μ_A, μ_B, μ_C, and μ_N are membership functions of sets A, B, C and N of the variables E_{Te}, $E_{\lambda s}$, φ and n, respectively.

Thus, the membership function μ_N of the output n is given by:

$$\mu_N(n) = \max_{i=1}^{72}\left(\mu'_{N_i}(n)\right) \tag{11}$$

We chose to share the universe of discourse of the stator flux error into two fuzzy sets, that of electromagnetic torque error in five and finally for the flux argument into seven fuzzy sets. This choice was based on Table 1. However the number of membership functions (fuzzy set) for each variable can be increased and therefore the accuracy is improved. All the membership functions of fuzzy controller are given in Fig. 1.

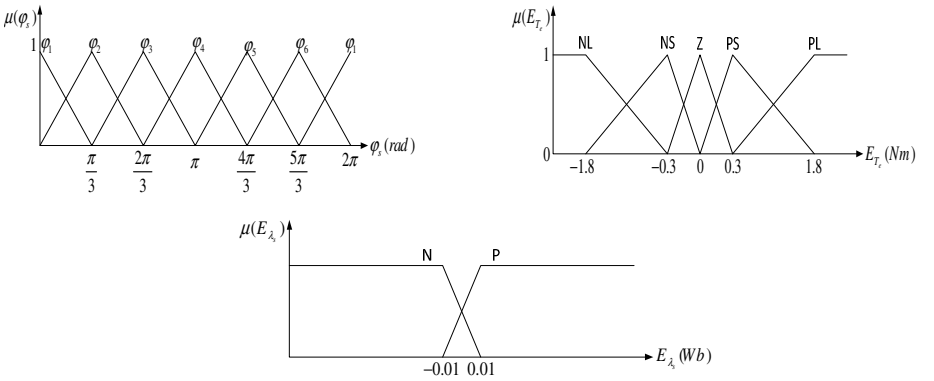


Fig. 1. Membership functions for fuzzy logic controller

Table 2. Fuzzy rules

		φ					
$E_{\lambda s}$	E_{Te}	φ_1	φ_2	φ_3	φ_4	φ_5	φ_6
	P	V_6	V_2	V_3	V_1	V_5	V_4
PL	Z	V_4	V_6	V_2	V_3	V_1	V_5
	N	V_5	V_4	V_6	V_2	V_3	V_1
	P	V_6	V_2	V_3	V_1	V_5	V_4
PS	Z	V_7	V_0	V_7	V_0	V_0	V_0
	N	V_5	V_4	V_6	V_2	V_3	V_1
	P	V_2	V_3	V_1	V_5	V_4	V_6
NS	Z	V_0	V_7	V_0	V_7	V_0	V_7
	N	V_1	V_5	V_4	V_6	V_2	V_3
	P	V_2	V_3	V_1	V_5	V_4	V_6
NL	Z	V_3	V_1	V_5	V_4	V_6	V_2
	N	V_1	V_5	V_4	V_6	V_2	V_3

4 Neural Network Based Direct Torque Control

This section presents the outline of neural networks to emulate the table of inverter switching states of DTC. The input signals of the table are the errors of electromagnetic torque, stator flux and the position vector of flux. The output signals are the inverter switching states n_a, n_b and n_c. As the switching table depends only on the electromagnetic torque error, stator flux angle and sector where the flux is located, and induction motor parameters, this neural network can be trained independently of the set. With the changes in the switching table reduces the training patterns and increases the execution speed of training process. This has been achieved by reducing the table to convert input analog signals to a digital bit for the flux error, two bits for the torque error and three bits for the flux position, which has a total of six inputs and three outputs, and only sixty-four training patterns. With these modifications, the network used to simulate has the advantage that it is independent of parameter variation of induction motor. This allows applying to any induction motor irrespective of its power.

From the flux space vectors λ_{ds} and λ_{qs} we can calculate the flux angle φ and flux magnitude λ_s. The coding of the flux angle is given by ξ_1, ξ_2 and ξ_3 according to following equations:

$$\lambda_s = \sqrt{\lambda_{ds}^2 + \lambda_{qs}^2}\ ,\ \varphi_s = \tan^{-1}\frac{\lambda_{ds}}{\lambda_{qs}},\ \xi_1\xi_2\xi_3 = encoder(\varphi_s) \tag{12}$$

$$\xi_1 = \begin{cases} 1 & \lambda_{qs} \geq 0 \\ 0 & otherwise \end{cases} \tag{13}$$

$$\xi_2 = \begin{cases} 1 & \left(\frac{\lambda_{qs}}{\lambda_{ds}} \geq -\tan(\frac{\pi}{3}) \quad and \quad \lambda_{ds} \prec 0\right) or \left(\frac{\lambda_{qs}}{\lambda_{ds}} \prec -\tan(\frac{\pi}{3}) \quad and \quad \lambda_{qs} \prec 0\right) \\ 0 & otherwise \end{cases} \tag{14}$$

$$\xi_3 = \begin{cases} 1 & \left(\frac{\lambda_{qs}}{\lambda_{ds}} \prec \tan(\frac{\pi}{3}) \quad and \quad \lambda_{ds} \geq 0\right) or \left(\frac{\lambda_{qs}}{\lambda_{ds}} \geq \tan(\frac{\pi}{3}) \quad and \quad \lambda_{qs} \prec 0\right) \\ 0 & otherwise \end{cases} \tag{15}$$

The network structure used, as shown in Fig. 2 has an input layer with five neurons, a first hidden layer with six neurons, a second hidden layer with five neurons and an output layer with three neurons. After training satisfactory, taking the weights and thresholds calculated and placed into the neural network prototype replacing the switching table. This network is incorporated as a part of the DTC.

5 Simulation Results

To compare and verify the proposed techniques in this paper, a digital simulation based on Matlab/Simulink program with a Neural Network Toolbox and Fuzzy Logic Toolbox is used to simulate the NN-DTC and FL-DTC, as shown in Fig. 4. The block diagram of a C-DTC/FL-DTC/NN-DTC controlled induction motor drive fed by a 2-level inverter is shown in Fig. 3. The induction motor used for the simulation studies has the following parameters:

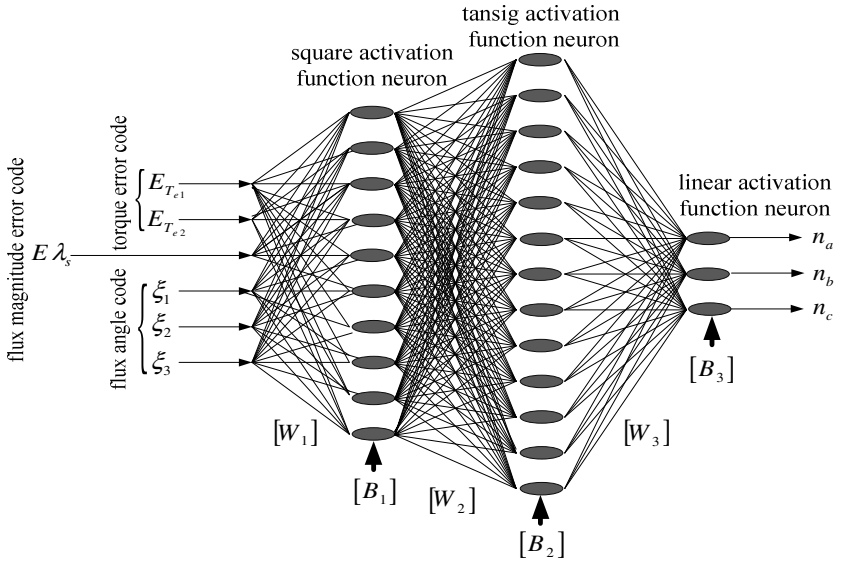


Fig. 2. Neural-network implementation of DTC

Rated power = 7.5kW, Rated voltage = 220V, Rated frequency = 60Hz, R_r = 0.17Ω, R_s = 0.15Ω, L_r = 0.035H, L_s = 0.035H, L_m = 0.0338H, J = 0.14kg.m^2.

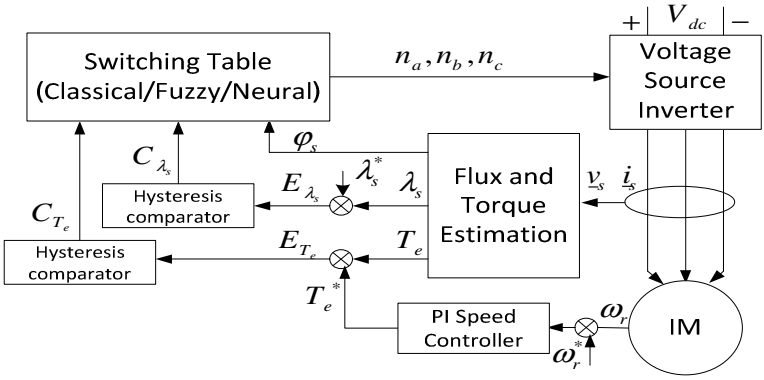
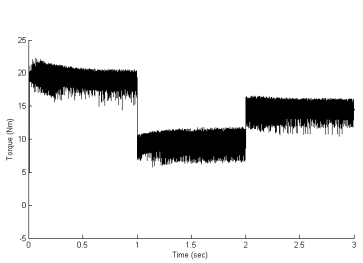
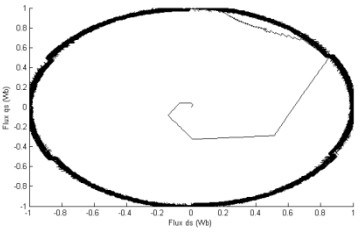


Fig. 3. General configuration of C-DTC/FL-DTC/NN-DTC scheme

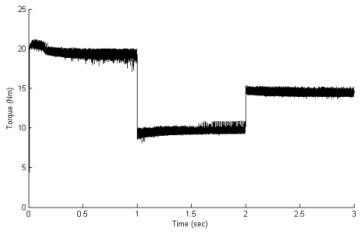
Figs. 4(a), 4(b) and 4(c) show the torque response of the C-DTC, FL-DTC and NN-DTC respectively with a torque reference of [20-10-15] Nm. While Figs. 4(a'), 4(b') and 4(c') show the flux response of the C-DTC, FL-DTC and NN-DTC respectively with a stator flux reference of 1Wb.



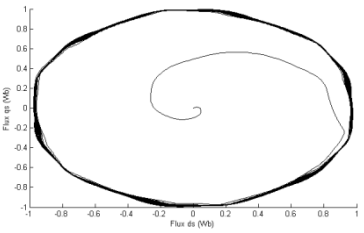
a



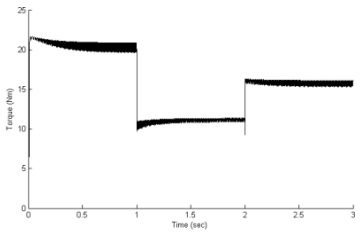
a'



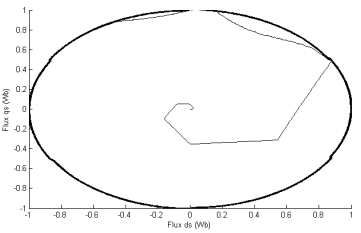
b



b'



c



c'

Fig. 4. (a), (b) and (c) torque response of C-DTC, FL-DTC and NN-DTC respectively, (a'), (b') and (c') Stator flux trajectory response of C-DTC, FL-DTC and NN-DTC respectively

Table 3 represents the comparative results in both stator flux and torque ripples percentage for C-DTC, FL-DTC and NN-DTC. The steady state response for the torque in NN-DTC is faster and provided more accuracy compared to other control strategies presented in this paper.

Table 3. Comparative study of C-DTC, FL-DTC and NN-DTC

Control strategies	Torque ripple (%)	Flux ripple (%)	Rise time (sec)	Setting time (sec)
C-DTC	10.6	2.3	0.009	0.01
FL-DTC	3.9	2.1	0.007	0.0085
NN-DTC	2.9	1.6	0.006	0.0082

6 Conclusions

A comparative study of C-DTC, FL-DTC and NN-DTC for an inverter fed induction motor drive was proposed in this paper. A better precision in the torque and flux responses was achieved with the NN-DTC method with greatly reduces the execution time of the controller; hence the steady-state control error is almost eliminated. The application of neural network techniques simplifies hardware implementation of direct torque control and it is envisaged that NN-DTC induction motor drives will gain wider acceptance in future.

References

1. Takahashi, I., Noguchi, T.: New Quick-Response and High-Efficiency Control Strategy of an Induction Motor. IEEE Transactions on Industry Applications IA-22(5), 820–827 (1986)
2. Noguchi, T., Yamamoto, M., Kondo, S., Takahashi, I.: Enlarging Switching Frequency in Direct Torque-Controlled Inverter by Means of Dithering. IEEE Transaction on Industry Applications 35(6), 1358–1366 (1999)
3. Depenbrock, M.: Direct Self-Control (DSC) of Inverter-Fed Induction Machine. IEEE Transaction on Power Electronics 3(4), 420–429 (1988)
4. Abdelli, R., Rekioua, D., Rekioua, T.: Performances Improvements and Torque Ripple Minimization for VSI Fed Induction Machine with Direct Control Torque. ISA Transactions 50(2), 213–219 (2011)
5. Vas, P.: Sensorless Vector and Direct Torque Control. University Press, London (1998)
6. Hassan, A.A., Shehata, E.G.: High Performance Direct Torque Control Schemes for an IPMSM Drive. Electric Power Systems Research 89, 171–182 (2012)
7. Wei, X., Chen, D., Zhao, C.: Minimization of Torque Ripple of Direct-Torque Controlled Induction Machines by Improved Discrete Space Vector Modulation. Electric Power Systems Research 72(2), 103–112 (2004)
8. Marino, R., Tomei, P., Verrelli, C.M.: Induction Motor Control Design. Springer London Ltd. (2010)
9. Livingstone, D.J.: Artificial Neural Networks: Methods and Applications. Humana Press Inc. (2009)
10. Chow, T.W.S., Cho, S.-Y.: Neural Networks and Computing: Learning Algorithms and Applications, Har/Cdr edn. Imperial College Press (2007)
11. Cong, W., Hill, D.J.: Learning From Neural Control. IEEE Transactions on Neural Networks 17(1), 130–146 (2006)

12. Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)
13. Buckley, J., Ying, H.: Expert Fuzzy Controller. Fuzzy Sets Syst. 43, 127–137 (1991)
14. Cintula, P., Hájek, P., Noguera, C. (eds.): Handbook of Mathematical Fuzzy Logic (in 2 volumes). Studies in Logic, Mathematical Logic and Foundations, vol. 37, 38. College Publications, London (2011)
15. Dvořák, A., Novák, V.: Formal Theories and Linguistic Descriptions. Fuzzy Sets and Systems 143(1), 169–188 (2004)
16. Kumsuwan, Y., Premrudeepreechacharn, S., Toliyat, H.A.: Modified Direct Torque Control Method for Induction Motor Drives based on Amplitude and Angle Control of Stator Flux 78, 1712–1718 (2012)
17. Yamamura, S.: Theory of the Linear Induction Motor. John Wiley & Sons (1972)
18. Mamdani, E.H., Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies 7(1), 1–13 (1975)
19. Zimmermann, H.J.: Fuzzy Sets, Decision Marking, and Expert Systems, Boston, Dordrecht, Lancaster (1987)

Classifier Ensembles Integration with Self-configuring Genetic Programming Algorithm

Maria Semenkina and Eugene Semenkin

Department of System Analysis and Operation Research, Siberian State Aerospace University, Krasnoyarsky Rabochy Avenue, 31, 660014, Krasnoyarsk, Russia
semenkina88@mail.ru, eugenesemenkin@yandex.ru

Abstract. Artificial neural networks and symbolic expression based ensembles are used for solving classification problems. Ensemble members and the ensembling method are generated automatically with the self-configuring genetic programming algorithm that does not need preliminary adjusting. Performance of the approach is demonstrated with real world problems. The proposed approach demonstrates results competitive to known techniques.

Keywords: self-configuring genetic programming, artificial neural networks, symbolic expressions, ensembles, classification problems.

1 Introduction

Classification is a well-known application of natural computing algorithms. Within machine learning domain, problems in which aim is to assign each input vector to one of a finite number of discrete categories are called classification problems [1]. The classification problem solving is usually described in terms of an optimization procedure that maximizes the number of correctly classified instances and minimizes the number of misclassified ones. This makes classification problems an appropriate area for the application of nature-inspired intellectual information processing technologies (IIT) like neural networks, fuzzy systems, evolutionary computations and many others.

The highly increasing computing power and technology made possible the use of more complex intelligent architectures, taking advantage of more than one intelligent system in a collaborative way. This is an effective combination of intelligent techniques that outperforms or competes to simple standard intelligent techniques.

One of the hybridization forms, the ensemble technique, has been applied in many real world problems. It has been observed that the diversity of members, making up a "committee", plays an important role in the ensemble approach [2-5]. Different techniques have been proposed for maintaining the diversity among members by running on the different feature sets [6] or training sets (e.g., bagging [7] and boosting [8]). Some techniques, such as neural networks, can be run on the same feature and training sets producing the diversity by different structures [9]. Simple averaging, weighted averaging, majority voting and ranking are common methods usually applied to calculate the ensemble output.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 60–69, 2013.

Johansson et al. [10] used genetic programming (GP) [11] for building an ensemble from the predefined number of the ANNs where the functional set consisted of the averaging and multiplying and the terminal set included the models (i.e., ANNs) and constants. In [12], a similar approach was proposed where first a specified number of the neural networks are generated and then a GP algorithm is applied to build an ensemble making up the symbolic regression function from partial decisions of the specific members.

In this paper, we apply the self-configuring genetic programming technique [13] to construct formula that shows how to compute an ensemble decision using the component decisions. The algorithm involves different operations and math functions and uses the models providing the diversity among the ensemble members. Namely, we use neural networks and symbolic expressions, automatically designed with our GP algorithm, as the ensemble members. The algorithm automatically chooses components which are important for obtaining an efficient solution and doesn't use the others.

With the approach developed an end user has no necessity to be an expert in the computational intelligence area but can implement the reliable and effective classification tool. It makes the approach very useful for different area experts making them free from extra efforts on the intellectual information technology algorithmic core implementation and allowing them to concentrate their attention in the area of their expertise, e.g. medicine, finance, engineering, etc.

The rest of the paper is organized as follows. Section 2 describes the method for the GP self-configuring and it's testing results confirming the method usefulness. Section 3 describes the method of the ANN automated design and its performance evaluation. In Section 4 we describe the GP-based approach to the IIT ensembles automated integration and the results of the performance comparative analysis on three relatively simple and two harder classification problems. In Section 5 we apply developed approach to hard real world problems solving from the area of the speech recognition and computer security. In Conclusion section we discuss the results and directions of the further research.

2 Operator Rates Based Self-configuration of GP Algorithm

Before suggesting the GP use to end users, e.g., medicine or finance specialists, for application in classification tools development, we have to save them from main troubles which are the problem even for evolutionary computation experts. It is really hard job to configure GP settings and tune its parameters, i.e., we have to suggest a way to avoid this problem.

We apply the operator probabilistic rates dynamic adaptation on the level of population with centralized control techniques [14-16]. To avoid the issues of precision caused while using real parameters, we used setting variants, namely types of selection, crossover, population control and level of mutation (medium, low, high). Each of these has its own probability distribution, e.g., there are 5 settings of selection: fitness proportional, rank-based, and tournament-based with three

tournament sizes. During initialization all probabilities are equal to 0.2 and they will be changed according to a special rule through the algorithm execution in such a way that the sum of probabilities should be equal to 1 and no probability could be less than predetermined minimum balance.

When the algorithm creates the next off-spring from the current population it first has to configure settings, i.e. to form the list of operators with using the probability operator distributions. The algorithm then selects parents with the chosen selection operator, produces an off-spring with the chosen crossover operator, mutates off-spring with the chosen mutation probability and puts off-spring into the intermediate population. When the intermediate population is filled, the fitness evaluation is computed and the operator rates (probabilities to be chosen) are updated according to operator productivities. Then the next parent population is formed. The algorithm stops after a given number of generations or if the termination criterion is met.

The productivity of an operator is the ratio of the average off-spring fitness obtained with this operator and the average fitness of the overall off-spring population. Winning operator increases its rate obtaining portions all other operators. We call our algorithm as self-configuring genetic programming (SelfCGP).

SelfCGP had demonstrated the high performance and reliability on benchmark symbolic regression problems with functions from [17] and in classification problems solving via symbolic expression based separating surfaces building [13]. It gives us a possibility to recommend SelfCGP for solving symbolic regression problems as better alternative to conventional GP. Main advantage of the SelfCGP is no need of algorithmic details adjustment without any losses in the performance that makes this algorithm useful for many applications where terminal users being no experts in evolutionary modelling intend to apply the GP for solving these problems.

3 ANN Automated Design with Self-configuring GP Algorithm

Usually, the GP algorithm works with tree representation, defined by functional and terminal sets, and exploit the specific solution transformation operators (selection, crossover, mutation, etc.) until termination condition will be met [11].

For the ANN automated design, the terminal set of our GP includes 16 activation functions such as bipolar sigmoid, unipolar sigmoid, Gaussian, threshold function, linear function, etc. The functional set includes specific operation for neuron placement and connections. The first operation is the placing a neuron or a group of neurons in one layer. There will no additional connections appeared in this case. The second operation is the placing a neuron or a group of neurons in sequential layers in such a way that the neuron (group of neurons) from the left branch of tree preceded by the neuron (group of neurons) from the right branch of tree. In this case, new connections will be added that connect the neurons from the left tree's branch with the neurons from the right tree's branch. Input neurons cannot receive any signal but have to send a signal to at least one hidden neuron.

The GP algorithm forms the tree from which the ANN structure is derived. The ANN training is executed to evaluate its fitness that depends on its performance in

solving problem in hand, e.g., the approximation precision or the number of misclassified instances. For training this ANN, connection weights are optimized with self-configuring genetic algorithm (SelfCGA) [18] that similarly to SelfCGP does not need any end user efforts to be the problem adjusted doing it automatically. When GP finishes giving the best found ANN structure as the result, this ANN is additionally trained with again SelfCGA hybridized with local search.

We compared the performance of the ANNs designed with our SelfCGP algorithm with the alternative methods on the set of problems from [19]. Materials for the comparison we have taken from [20] where together with results of authors' algorithm (CROANN) the results of 15 other approaches are presented on three classification problems (Iris, Wisconsin Breast Cancer, Pima Indian Diabetes) from [19].

Analysing comparison results, we observed ([21]) that the performance of our approach is high enough comparing to alternative algorithms (1st, 3rd and 4th positions, correspondingly). However, the main benefit from our SelfCGP algorithm is the possibility to be used by the end user without expert knowledge in ANN modelling and evolutionary algorithm application. Additional dividend is the size of designed ANNs. The ANNs designed with SelfCGP contain few hidden neurons and connections and use usually not all given inputs although perform well.

Now we can conclude that the self-configuring genetic programming algorithm is the suitable tool for ANN automated design.

4 Integration of IIT Ensembles with Self-configuring Genetic Programming Algorithm

Having developed appropriate tool for IIT automated design that does not require the effort for its adjustment, we applied our self-configuring genetic programming technique to construct formula that shows how to compute an ensemble decision using the component IIT decisions. The algorithm involves different operations and math functions and uses the models of different kinds in terminal set that provides the diversity among the ensemble members. In our numerical experiments, we use symbolic expressions and neural networks, automatically designed with our SelfCGP algorithm, as the ensemble members. The algorithm automatically chooses the component IIT which are important for obtaining an efficient solution and doesn't use the others. The ensemble component IIT are taken from the preliminary IIT pool that includes 10 ANNs and 10 symbolic regression formulas (SRFs) generated in advance with SelfCGP. For the designing every IIT, corresponding data set is randomly divided into two parts, i.e., training sample (70%) and test sample (30%).

The first experiment was conducted for comparing the performance of the ensembling method based on the SelfCGP with the others, i.e. simple averaging, weighted averaging and bagging [7]. Bagging technique assumes that all members of the ensemble are trained on different subsets of the training sample. In Table 1 below, "ensemble+bagging" means that ensemble members were generated with SelfCGP using different subsets of the training sample and then corresponding ensemble was designed with SelfCGP. We used the same three problems from [19] as the test bed.

In Table 1 below we present our results. Numbers in columns are the error measure calculated as it was given in [20]. The first six lines contain results of the ensembling method suggested in this paper (with bagging and without it) for three kinds of ensemble members – ANNs, SRFs and both. Next eight lines contain results for conventional methods of ensemble forming, i.e., conventional bagging, simple and weighted averaging. Next two lines shows results of single technologies (ANN and SRF) automatically generated with SelfCGP. Results of these lines are averaged over 20 independent runs. Results for the last 15 lines were taken from [20] for the comparison.

Table 1. Ensembling methods comparison

Classifier	Iris	Cancer	Diabetes
SelfCGP+ANN+Ensemble+Bagging	0	0	17.13
SelfCGP+ANN+Ensemble	0	0	17.18
SelfCGP+ANN+SRF+Ensemble+Bagging	0	0	17.41
SelfCGP+ANN+SRF+Ensemble	0	0.06	17.43
SelfCGP+SRF+Ensemble+Bagging	1.12	0.06	18.11
SelfCGP+SRF+Ensemble	1.33	0.34	18.21
Bagging (SelfCGP+ANN)	1.26	0.67	18.22
Bagging (SelfCGP+SRF)	2.67	0.95	19.34
ANN ensemble with weighted averaging	2.67	1.03	19.03
ANN ensemble with simple averaging	2.67	1.09	19.75
SRF ensemble with weighted averaging	4.00	1.22	19.86
SRF ensemble with simple averaging	5.33	1.27	20.23
ANN+SRF ensemble with weighted averaging	2.67	1.09	19.34
ANN+SRF ensemble with simple averaging	4.00	1.18	19.79
SelfCGP+ANN	1.33	1.05	19.69
SelfCGP+SRF	2.67	1.23	20.01
CROANN	1.31	1.06	19.67
GANet-best	6.40	1.06	24.70
SVM-best	1.40	3.10	22.70
CCSS	4.40	2.72	24.02
COOP	-	1.23	19.69
CNNE	-	1.20	19.60
EPNet	-	1.38	22.38
EDTs	-	2.63	-
SGAANN	14.20	1.50	24.46
EPANN	12.56	1.54	25.75
ESANN	7.08	0.95	20.93
PSOANN	10.38	1.24	20.99
GSOANN	3.52	0.65	19.79
MGNN	4.68	3.05	-
EENCL	-	-	22.1

Results in Table 1 demonstrate that the SelfCGP based ensembling method used the ANNs or ANNs and SRFs integration outperforms conventional ensembling methods, the single best ANN and SRF designed with SelfCGP as well as other given classification methods. We can also see that bagging does not produce impact being added to our ensembling technique based on ANNs. The statistical robustness of the results obtained was confirmed by ANOVA tests which were used for processing received evaluations of our algorithms performance.

Within the second numerical experiment we solved two hard classification problems and compared our results with alternative approaches. These problems are so called German and Australian Credit Data Sets from the UCI Repository of Machine Learning Databases [19], and are often used to compare the accuracy with various classification models.

Results for alternative approaches have been taken from scientific literature. In [22] the performance evaluation results for these two data sets are given for authors' two-stage genetic programming algorithm (2SGP) as well as for the following approaches taken from other papers: conventional genetic programming (GP+SRF), classification and regression tree (CART), C4.5 decision trees, k nearest neighbors (k-NN), linear regression (LR). Additional material for comparison we have taken from [23] where is evaluation data for authors' automatically designed fuzzy rule based classifier as well as for other approaches found in literature: Bayesian approach, boosting, bagging, random subspace method (RSM), cooperative coevolution ensemble learning (CCEL).

The results of the comparison (the proportion of correctly classified objects in the test set) are given in Table 2 where the last letter "E" means "ensemble".

Table 2. Performance comparison for bank credit scoring problems

Classifier	Australian credit	German credit	Classifier	Australian credit	German credit
SelfCGP ANN+SRFE	0.9094	0.8126	Fuzzy	0.8910	0.7940
SelfCGP ANN+SRFE+Bag	0.9093	0.8127	2SGP	0.9027	0.8015
SelfCGP ANNE+Bag	0.9092	0.8125	GP+SRF	0.8889	0.7834
SelfCGP SRFE+Bag	0.9071	0.8050	LR	0.8696	0.7837
Bagging (SelfCGP+ANN)	0.9071	0.8004	Bayesian	0.8470	0.6790
SelfCGP ANNE	0.9046	0.8075	RSM	0.8660	0.7460
SelfCGP SRFE	0.9046	0.8050	k-NN	0.8744	0.7565
Bagging (SelfCGP+SRF)	0.9046	0.7975	CART	0.8986	0.7618
SelfCGP+SRF	0.9022	0.7950	C4.5	0.8986	0.7773
SelfCGP+ANN	0.9022	0.7954	CCEL	0.7150	0.7151
GP+ANN	0.8969	0.7863	Bagging	0.8470	0.6840
			Boosting	0.7600	0.7000

As one can see, the ensembles automatically designed with the SelfCGP outperform other ensembles (Boosting, Bagging, CCEL) and single classifiers including these specially implemented for bank scoring problems solving (Fuzzy, 2SGP). We can also see that bagging here does not bring improvement if it is added to heterogeneous ensemble (ANN+SRF) but it brings higher performance being added to homogenous ensembles consisted of ANN or SRF.

5 IIT Ensemble Design for Solving Computing Technologies Problems

Successful application of our approach in the area of classification brought us to the idea of its adapting to the complex problems from the area of computing technologies such as speech recognition and computing security.

In the area of speech recognition we have chosen the ISOLET problem ([19]) due to the availability of the other approaches known results for the comparison.

ISOLET problem is the recognition problem of English letters pronounced by 150 different speakers those spoke the name of each letter of the alphabet twice. The features include spectral coefficients; contour features, sonorant features, pre-sonorant features, and post-sonorant features. Exact order of appearance of the features is not known. It gives the data set with 617 attributes (all of them are continuous, real valued attributes scaled into the range -1.0 to 1.0), 26 classes, and 7797 instances.

Having in mind the necessity to verify the ensembles, we have randomly divided the data set in three parts: 4679 instances for single ANNs training, 1559 instances for single ANNs testing and 1559 instances for the ensembles cross-validation. Ensembles training were executed on the first 6238 instances. Both ANN-based classifiers and their ensembles were automatically designed with SelfCGP algorithm. Preliminary pool of classifiers consisted of 10 members. SRFs are not used here because of the problem complexity (26 classes).

Alternative approaches for performance comparison have been taken from [19]. In table 3 bellow OPT means conjugate-gradient implementation of back propagation, C4.5 means Quinlan's C4.5 system, OPC means one-per-class representation, ECOC means error-correcting output code.

As one can see from Table 3 where we show first 12 of 41 lines, both our approaches demonstrate competitive results (2*nd* and 10*th* position of 41). ANN-based classifier automatically generated with SelfCGP can be considered as enough powerful tool for speech recognition problems but our ensembling technique can essentially improve the classifier performance making it to be one of the best among competitors.

Table 3. Performance comparison for ISOLET problem

Algorithms and their configurations	% errors	% correct
OPT 30-bit ECOC	3.27	96.73
SelfCGP+ANN+Ensemble	3.40	96.60
SelfCGP+ANN+Ensemble+Bagging	3.40	96.60
OPT 62-bit ECOC	4.04	95.96
Bagging (SelfCGP+ANN)	4.17	95.93
OPT OPC	4.17	95.83
C4.5 107-bit ECOC soft pruned	6.61	93.39
C4.5 92-bit ECOC soft pruned	6.86	93.14
C4.5 45-bit ECOC soft pruned	6.99	93.01
SelfCGP+ANN	7.21	92.79
C4.5 107-bit ECOC soft raw	7.44	92.56
C4.5 92-bit ECOC soft raw	7.57	92.43

Here bagging does not improve the performance of our homogenous ANN-based ensemble although, of course, it improves the performance of single ANN generated with SelfCGP.

In the area of computer security we have chosen the problem of the detection of PROBE attacks. Corresponding dataset "KDD'99 Cup" is hosted in the Machine Learning Repository [19]. For the approach effectiveness evaluation, all patterns relevant to PROBE attacks were marked as referring to the first class, the others were marked as belonging to the second class. We used the following attributes in our experimental study: 1, 3, 5, 8, 33, 35, 37, 40. The choice of these attributes has been made empirically based on the analysis of related works and their description can be found in [24]. The results were compared with other approaches collected in [25]. The comparison results are shown in Table 4 below.

Table 4. Performance comparison for PROBE attack detectors

Applied technique	Detection rate, %	False positive rate, %
PSO-RF	**99.92**	**0.029**
SelfCGP+ANN+Ensemble+Bagging	**99.80**	**0.028**
SelfCGP+ANN+Ensemble	**99.79**	**0.027**
Random Forest	**99.80**	0.100
Bagging (SelfCGP+ANN)	99.24	0.078
SelfCGP+ANN	98.78	0.097
Bagging	99.60	0.100
PART (C4.5)	99.60	0.100
NBTree	99.60	0.100
Jrip	99.50	0.100
Ensemble with majority voting	99.41	0.043
Ensemble with weighted averaging	99.17	0.078
Ensemble with simple averaging	99.18	0.122
BayesNet	98.50	1.000
SMO (SVM)	84.30	3.800
Logistic	84.30	3.400

From Table 4 we can conclude that the classifier automatically designed with SelfCGP as the ANN-based ensemble demonstrates the high performance compared with the best known results (PSO-RF and RF). The classifier based on the single ANN designed with SelfCGP also demonstrates competitive performance. Bagging again does not improve the ensemble performance although improves it for ANN.

6 Conclusion

The SelfCGP based automatic design of IITs ensembles allows improving the effectiveness of classification. Obtained results are approved by solving some hard real-world problems.

Comparing with usual ensembling methods, e.g. weighted averaging or voting, additional computational effort for our approach is the necessity to run the genetic programming algorithm that combines single models outputs into an output of the ensemble that could be considered as the acceptable disadvantage.

As about the model complexity, of course, a computational model given by the genetic programming algorithm could be much more complicated comparing to usual ensembling methods but in our experiments we did not observe great complexity of ensembles. Examples of usual ensembles are $cos(N_0)+0.5 \cdot N_8 \cdot N_9$ for Australian credit and $1-(1-N_0) \cdot (1-N_1)$ for German credit. Here N_i are outputs of automatically designed ANNs. Our experiments show that self-configuring genetic programming algorithm never includes all available single models into an ensemble taking usually a few of them. As the greater part of the ensemble computational complexity is given by the computational efforts needed for calculating the output for each model, our approach has the advantage upon usual ensembling methods that usually include all available single models in the ensemble as well as upon SelfCGP based bagging approach that usually requires two or three times more ensemble members designed with SelfCGP (e.g., $N_0 \cdot \sin(1.53 \cdot N_2 \cdot N_5) \cdot \sqrt{N_9 \cdot N_7} + 0.72 \cdot N_0 \cdot N_3 - 0.69 \cdot N_6$ for German credit) and doesn't improve the ensemble performance.

The further development of our approach is aimed to the expansion of its functionality by including the other types of IITs (fuzzy logic systems, decision trees, neuro-fuzzy systems, other kinds of ANNs, etc.).

Acknowledgments. The research is partially supported through the Governmental contracts № 16.740.11.0742 and 11.519.11.4002.

References

1. Bishop, C.: Pattern recognition and machine learning. Springer (2006)
2. Dietterich, T.G.: An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Machine Learning 40(2), 139–158 (2000)
3. De Stefano, C., Della Cioppa, A., Marcelli, A.: An evolutionary approach for dynamic configuration of multi-expert classification systems. In: Proceedings IEEE Congress on Evolutionary Computation, CEC 2006, pp. 2444–2450 (2006)
4. Kim, Y.W., Oh, I.S.: Classifier ensemble selection using hybrid genetic algorithms. Pattern Recognition Letters 29(6), 796–802 (2008)
5. Rokach, L.: Ensemble-based classifiers. Artificial Intelligence Review 33(1), 1–39 (2010)
6. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(1), 66–75 (1994)
7. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
8. Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics 28(2), 337–374 (2000)
9. Navone, H.D., Granitto, P.M., Verdes, P.F., Ceccatto, H.A.: A learning algorithm for neural network ensembles. Inteligencia Artificial, RevistaIberoamericana de Inteligencia Artificial (12), 70–74 (2001)

10. Johansson, U., Lofstrom, T., Konig, R., Niklasson, L.: Building Neural Network Ensembles using Genetic Programming. In: International Joint Conference on Neural Networks (2006)
11. Poli R., Langdon W.B., McPhee N.F.: A Field Guide to Genetic Programming (2008), Published via, http://lulu.com and freely available, http://www.gp-field-guide.org.uk
12. Bukhtoyarov, V., Semenkina, O.: Comprehensive evolutionary approach for neural network ensemble automatic design. In: Proceedings of the IEEE World Congress on Computational Intelligence, pp. 1640–1645 (2010)
13. Semenkin, E., Semenkina, M.: Self-Configuring Genetic Programming Algorithm with Modified Uniform Crossover. In: Liu, J., et al (eds.): CEC IEEE WCCI 2012, Congress on Evolutionary Computations of IEEE World Congress on Computational Intelligence, Brisbane, Australia, pp. 1918–1923 (2012)
14. Gomez, J.: Self Adaptation of Operator Rates in Evolutionary Algorithms. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 1162–1173. Springer, Heidelberg (2004)
15. Meyer-Nieberg, S., Beyer, H.-G.: Self-Adaptation in Evolutionary Algorithms. In: Lobo, F., Lima, C., Michalewicz, Z. (eds.) Parameter Setting in Evolutionary Algorithm, pp. 47–75 (2007)
16. Angeline, P.J.: Two Self-Adaptive Crossover Operators for Genetic Programming. In: Angeline, P.J., Kinnear Jr., K.E. (eds.) Advances in Genetic Programming, vol. 2, pp. 89–110 (1996)
17. Finck, S., et al.: Real-parameter black-box optimization benchmarking 2009. In: Presentation of the noiseless functions. Technical Report ResearhCenter PPE (2009)
18. Semenkin, E., Semenkina, M.: Self-configuring Genetic Algorithm with Modified Uniform Crossover Operator. In: Tan, Y., Shi, Y., Ji, Z. (eds.) ICSI 2012, Part I. LNCS, vol. 7331, pp. 414–421. Springer, Heidelberg (2012)
19. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2010), http://archive.ics.uci.edu/ml
20. Yu, J.J.Q., Lam, A.Y.S., Li, V.O.K.: Evolutionary Artificial Neural Network Based on Chemical Reaction Optimization. In: IEEE Congress on Evolutionary Computation (CEC 2011), New Orleans, LA (2011)
21. Semenkin, E.S., Semenkina, M.E.: Artificial neural networks design with self-configuring genetic programming algorithm. In: Filipic, B., Silc, J. (eds.) Bioinspired Optimization Methods and their Applications: Proceedings of the Fifth International Conference, BIOMA 2012, pp. 291–300. Jozef Stefan Institute, Ljubljana (2012)
22. Huang, J.-J., Tzeng, G.-H., Ong, C.-S.: Two-stage genetic programming (2SGP) for the credit scoring model. Applied Mathematics and Computation 174, 1039–1053 (2006)
23. Sergienko, R., Semenkin, E., Bukhtoyarov, V.: Michigan and Pittsburgh Methods Combining for Fuzzy Classifier Generating with Coevolutionary Algorithm for Strategy Adaptation. In: 2011 IEEE Congress on Evolutionary Computation, New Orleans, LA (2011)
24. Stolfo, S., Fan, W., Lee, W., Prodromidis, A., Chan, P.: Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. In: Proceedings of the 2000 DARPA In-formation Survivability Conference and Exposition, DISCEX 2000 (2000)
25. Malik, A.J., Shahzad, W., Khan, F.A.: Binary PSO and random forests algorithm for PROBE attacks detection in a network. In: IEEE Congress on Evolutionary Computation, pp. 662–668 (2011)

A Multi-objective Proposal Based on Firefly Behaviour for Green Scheduling in Grid Systems

María Arsuaga-Ríos[1] and Miguel A. Vega-Rodríguez[2]

[1] Beams Department, European Organization for Nuclear Research, CERN.
1211 Geneva 23, Switzerland
maria.arsuaga.rios@cern.ch
[2] ARCO Research Group, University of Extremadura,
Dept. Technologies of Computers and Communications,
Escuela Politécnica, Cáceres, Spain
mavega@unex.es

Abstract. Global warming and climate change are threats that the planet is facing nowadays. Green computing has emerged as a challenge to reduce the energy consumption and pollution footprints of computers. Grid Computing could match the principles of Green Computing as it could exploit and efficiently use processors' computing power. This paper presents a swarm multi-objective optimization algorithm for scheduling experiments (the job execution) on the Grid. Multi-Objective Firefly Algorithm (MO-FA) is inspired by the brightness attraction among fireflies. One of the main contributions of this work is that the increasing firefly brightness is interpreted as an improvement in response time and energy savings. This would fulfill both conflicting objectives of Grid users: execution time and energy consumption. Results show that MO-FA is a reliable method according to its interquartile range and its comparison with the standard and well-known multi-objective algorithm NSGA-II. Moreover, it performs better than actual grid schedulers as the Workload Management System (WMS) and the Deadline Budget Constraint (DBC).

Keywords: swarm, multi-objective optimization, green computing, grid environment, scheduling.

1 Introduction

Green Computing also called Green IT has emerged to accomplish the efficiency of computational resources. Its aims are: environment impact minimization and economic availability maximization. At present, a fast advance is occurring in the creation of energy-efficient supercomputers, in fact there is a world ranking list called *The Green 500*[1]. One of the infrastructures that could contribute reducing the energy consumption or the carbon emission is Grid Computing[2].

[1] http://www.green500.org/
[2] http://www.gridtalk.org/Documents/gridsandgreen.pdf

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 70–79, 2013.

Grid computing gives access to geographically and heterogeneous distributed resources using their power as a big and powerful processor [8]. Job scheduling is already a non-trivial problem widely studied to optimize the response time for experiments submitted on Grid environments ([7], [15], [21]). They tried to fulfill deadlines or at least minimize the execution time for scientific applications with dependent or independent jobs.

However, the progress of green computing is opening a new goal for this problem. Current researches are focused on the energy management taking into account the energy consumption on idle resources or nodes ([2], [9], [5], [18]), even cores of processing for multiprocessors [14], or just considering the energy reduction for communication data [17]. Most of these approaches use heuristics based on the Dynamic Voltage and Frequency Scaling (DVFS) technique [23]. They try to optimize the use of energy by decreasing the voltage and the clock frequency (CPU speed) on the idle nodes or nodes that are executing non-critical jobs. These researches reduce the energy consumption by heuristics or greedy algorithms and some of them try to balance this reduction regarding the execution time by using a single objective function with weights for evaluating these objectives. In this paper, a new approach is studied related to the multi-objective optimization for these two conflictive objectives, considering them separately and with the same importance. Khan's work ([10], [11], [12], [16]) takes into account this scheduling problem in a low level by simulating the power off and DVFS techniques for the machines. However, most of these simulated researches lack of detailed configuration for their topologies, they do not specify the characteristics for the used resources or the scheduling workflows are not described in detail. We try to solve all these lacks.

In addition, we include a comparison with real grid schedulers as the *Workload Management System* (WMS)[3] from the most used European middleware Lightweight Middleware for Grid Computing (gLite)[4] and also the well-known *Deadline Budget Constraint* (DBC) [3] from Nimrod-G.

In the present paper, we propose a novel swarm algorithm with a multi-objective approach to deal with the optimization of both critical goals - energy consumption and execution time - considering them with the same importance. Multi-objective Firefly Algorithm (MO-FA) is based on the mono-objective counterpart FA [24] inspired on the fireflies' behaviour. The main feature of this algorithm is the communication between the fireflies combining multi-objective exploitation and exploration processes to solve the problem. MO-FA is implemented in the simulator GridSim[5] which allows the configuration of complex topologies and the specification of resource characteristics such as processing speed, MIPS (Millions of Instructions Per Second) or power per time unit. In addition, several workflows have been used as Gaussian, Gauss-Jordan or LU Decomposition following a DAG (Directed Acyclic Graph) Model to support dependent jobs. This research contribution assumes the utilization of ecological resources that are emerging nowadays

[3] `http://web.infn.it/gLiteWMS/`
[4] `http://glite.cern.ch/`
[5] `http://www.buyya.com/gridsim/`

by using software and techniques as CLUES [6] or EnergySaving Cluster (ESC) [6]. These techniques manage idle resources and other approaches per site or machine. Therefore, our proposal is the perfect complement to this ecological software that is emerging. A very preliminary version of this work, with only two pages, was published in [1]. In this paper, this work has been considerably extended and improved. This paper is structured as follows. Section 2 defines the problem statement. Section 3 describes the MO-FA approach. Section 4 shows the test environment and the experimental results. Finally, Section 5 summarizes the conclusions.

2 Problem Statement

Grid scheduling consists in the job allocation on grid resources fulfilling the user requirements. In this research, we focus on two critical objectives: energy consumption and execution time. Nowadays, the reduction of energy consumption is an up-to-date goal as a result of the big importance of green computing, being execution time an essential issue in scheduling problems. Furthermore, these objectives are conflicting with each other due to faster resources frequently implies higher energy consumptions. Therefore, a multi-objective approach is necessary to tackle this problem. Multi-objective Optimization Problems (MOPs) [13] include a set of n parameters (decision variables) and a set of k objective functions. The objective functions are functions from the decision variables. Hence, a MOP could be defined as: $Optimize\ y = f(x) = (f_1(x), f_2(x), ..., f_k(x))$, where $x = (x_1, x_1, ..., x_n) \in X$ is the decision vector and $y = (y_1, y_2, ..., y_k) \in Y$ the objective vector. The decision space is denoted by X and Y is the objective space.

MOPs generally return a set of solutions. The set of optimum solutions is called Pareto optimal set and the point set, defined by the Pareto optimal set in the value space of the objective functions, is known as Pareto front. For a given MOP and Pareto optimal set P*, the Pareto front (PF) is defined as: $PF^* := \boldsymbol{f} = (f_1(x), f_2(x), , f_k(x))|x\epsilon P^*$. Pareto front consists just in non-dominated solutions. One solution dominates other if and only if, it is at least as good as the other in all the objectives and it is better in at least one of them.

Our Multi-objective Optimization Problem minimizes at the same time and with the same importance two critical objectives - energy consumption and execution time -. Given a set of grid resources $R = \{R_j\}$, j = 1,..,n and a set of jobs $J = \{J_i\}$, i = 1,..,m the fitness functions are defined as:

$$Min\ F = (F_1,\ F_2) \tag{1}$$

$$F_1 = \sum EnergyConsumption\,(J_i,\ f_j(J_i)) \tag{2}$$

$$F_2 = MaxTime\,(J_i,\ f_j(J_i)) \tag{3}$$

[6] http://www.grycap.upv.es/clues/eng/index.php

where $f_j(J_i)$ denotes the job J_i allocation on the resource R_j. The objective function F_1 returns the energy consumption for processing the experiment (set of jobs) and the objective function F_2 reports its completion time.

Experiments often are built up from workflows - set of jobs with dependencies among them - . Dependent jobs are critical for minimizing the execution time, because they need to wait to the successful execution of the predecessor jobs. Therefore, a Directed Acyclic Graph (DAG) model has been considered to represent the experiments submitted on Grid. A workflow is modeled by a weighted directed acyclic graph (DAG) $JG = (V, E, l, d)$, where V is a set of nodes and E is a set of edges. Each node $j \in V$ corresponds to a job and it has assigned a constant length measured in thousands of MI (Millions of Instructions), this length is denoted by $l(j)$. Each edge $(j \rightarrow j') \in E$ from j to j' denotes the dependency between the job j' regarding the job j. Job j' could not be executed until job j has been executed successfully. The transferred data length $d(j \rightarrow j')$ between the jobs is specified and measured in bytes.

3 Multi-Objective Firefly Algorithm (MO-FA)

Multi-objective Firefly Algorithm (MO-FA) is a new version of the original and mono-objective Firefly Algorithm (FA) [24] to support the optimization for more than one objective. FA is a swarm algorithm based on the fireflies behaviour. The outstanding characteristic of the fireflies is the attractiveness among them. This attractiveness is proportional to their brightness, therefore for any two flashing fireflies, the less bright one will move towards the brighter one. In order to apply this behaviour in optimization problems, the brightness represents the value of the fitness function. The multi-objective approach has to consider more than one fitness function and for solving our minimization problem the brightness attribute would be inversely proportional to the fitness functions. Following, MO-FA is described in Algorithm 1.

Algorithm 1 MO-FA pseudocode

INPUT: Population Size, β, γ, α
OUTPUT: Set of Solutions

```
1: Initialize population of solutions;
2: Evaluate population (Energy and Time);
3: while not stop condition do
4:     Multiobjective Comparison among fireflies;
5:     Multiobjective firefly attraction;
6: end while
7: Select Set of Best Solutions (First Pareto Front);
```

This algorithm requires four parameters: population size, β, γ and α. MO-FA considers the population size as the number of fireflies, which represent the candidate solutions. The initial attractiveness between two fireflies is denoted

by β. The coefficient of light absorption is indicated by γ and it represents the light absorbed from the air. And α is the randomization parameter. These three parameters are used in equation 5 to diversify the optimization process.

Each firefly represents a candidate solution. Two vectors are built for any firefly: allocation and order vector. Order vector denotes execution order of the jobs that compound the workflow. Allocation vector indicates the assignment between jobs and resources, where jobs will be executed. The combination of these two vectors conforms the firefly dimensional position called U_{O+A}. Energy consumption and execution time are the fitness values for each firefly returned by GridSim according to the previous vectors. This representation is based on the work [20]. The algorithm starts with a random initialization of the fireflies' population taking into account the dependencies among jobs. When the fireflies are generated, GridSim returns the values of energy consumption in watts and the execution time in seconds.

Then, each firefly is compared with the others in order to detect if there is any firefly with more brightness (better fitness values). This comparison is carried out per each pair of fireflies according to the dominance concept. The dominated firefly will move towards the firefly with better fitness values. The firefly attraction process needs first the calculation of the Euclidean distance. Euclidean distance is calculated from the U_{O+A} vectors of the pair of agents. Using in the first part of the vector (order vector), the distance between order positions and for the second part (allocation vector) the distance between the resource numbers (equation 4).

$$r_{i,j} = \|firefly_i - firefly_j\| = \sqrt{\sum_{k=1}^{dimension} (firefly_{i,k} - firefly_{j,k})^2} \qquad (4)$$

The movement of a firefly $firefly_i$ attracted to another more attractive (brighter) firefly $firefly_j$ is determined by equation 5 (applied to every dimension of the fireflies).

$$firefly_i = firefly_i + \beta e^{-\gamma r_{i,j}^2}(firefly_j - firefly_i) + \alpha(rand - \frac{1}{2}) \qquad (5)$$

where β indicates the attractiveness for $r = 0$, γ is the coefficient of light absorption and α is the randomization parameter, because *rand* denotes a random number between 0 and 1. This stochastic feature is considered to apply the exploration processes for this algorithm. In swarm algorithms, exploration and exploitation processes are required to avoid local optima and achieve good solutions.

Once all the fireflies are compared and the updated movements are carried out, a stagnation checking method is applied to the new firefly population. This method is an improvement with respect to the original FA in order to avoid the population stagnation during several iterations. Two mutation methods (one per each vector) are applied to the stagnated fireflies. Both mutations consider heuristics from the job scheduling problem carrying out a local search. Order and

allocation vectors are generated randomly but each one has its own heuristics. Random order vector is compared with other order vector built from a greedy algorithm. The greedy algorithm consists in the creation of an order vector where its first positions are assigned for the jobs that have more jobs dependents on it. This fact implies that these jobs will be executed first. The mutation for the allocation vector is more complex, since it considers more heuristics such as the first selection of the resources according to their speed/energy consumption, time per job in the resource selected, wait time taking into account the dependencies between jobs and the overhead time to predict the total execution of the workflow.

Then, the new population is processed again until the time limit is expired. After that, the fireflies are ranked by using the multi-objective operator *Pareto fronts classification* from the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [4]. Finally, the *first Pareto front* is extracted from the final population being the resulting set of solutions for the problem.

4 Test Environment and Experiments

A detailed grid topology and workflows are configured in GridSim to simulate a real grid behavior. The EU Data Grid testbed [19] is completed with the grid resources characteristics from the WWG tested [3] (Table 1). Moreover, we have simulated 3 working nodes (WNs) per resource to add more complexity to this environment. Three different and parallel numerical computation workflows are launched on this infrastructure: Parallel Gaussian Algorithm, Parallel Gauss-Jordan Algorithm and Parallel LU decomposition [22]. All of these workflows follow a DAG model with their respective lengths in thousands of MI (Millions of Instructions) and the input/output sizes in bytes.

Table 1. Resource Characteristics

Resource Name	Features (Vendor, Type, OS, CPUs/WN)	Resource Manager Type	MIPS /CPU	Power(W) /CPU time
LYON	Compaq, AlphaServer, OSF1, 4	Time-shared	515	67
CERN	Sun, Ultra, Solaris, 4	Time-shared	377	50
RAL	Sun, Ultra, Solaris, 4	Time-shared	377	50
IMPERIAL	Sun, Ultra, Solaris, 2	Time-shared	377	50
NORDUGRID	Intel, Pentium/VC820, Linux, 2	Time-shared	380	29.05
NIKHEF	SGI, Origin 3200, IRIX, 6	Time-shared	410	17
PADOVA	SGI, Origin 3200, IRIX, 16	Time-shared	410	17
BOLOGNA	SGI, Origin 3200, IRIX, 6	Space-shared	410	17
ROME	Intel, Pentium/VC820, Linux, 2	Time-shared	380	29.05
TORINO	SGI, Origin 3200, IRIX, 4	Time-shared	410	17
MILANO	Sun, Ultra, Solaris, 8	Time-shared	377	50

Experiments have been divided in two complementary studies to evaluate the feasibility of the proposed algorithm MO-FA. Due to the stochastic nature of

multi-objective metaheuristics, each experiment performed in our study includes 30 independent executions. In addition, other study for parameter settings was carried out in order to obtain the more suitable values for the tackled problem. This study was based on 30 independent executions per parameter value (at least 5 different equidistant values were checked for every parameter), considering the others as a constant. The final settings used are: Population size = 100, β= 0.2, $\gamma = 1$, $\alpha = 1$ and stop condition = 2 minutes.

On the one hand, the MO-FA behaviour is studied as a multi-objective algorithm compared with the standard and well-known multi-objective algorithm NSGA-II (Non-dominated Sorting Genetic Algorithm-II)[4] to evaluate the multi-objective properties and assess its reliability. Hypervolume [25] is a well-known performance measure used in multi-objective algorithms to evaluate the optimization regarding the solutions found. The hypervolume median, interquartile range and the maximum reference point (the other reference point is (0,0) in all the cases) are presented to consider them in future comparisons. Results shown in Table 2 demonstrate that its median of hypervolume percentages arises more than 50%, being a good point regarding the solutions found (MO-FA obtains better hypervolume values than NSGA-II). Moreover, the comparison with NSGA-II indicates the reliability of the solutions found. Set coverage metrics [25] is used to indicate the percentage of dominance among the solutions found per each algorithm. In Table 3, each cell gives the fraction of non-dominated solutions evolved by algorithm B, which are covered by the non-dominated points achieved by algorithm A [25]. Table 3 shows that MO-FA solutions dominate the solutions found by NSGA-II in all the workflows.

Table 2. Hypervolume properties per each workflow and algorithm

Workflows	MO-FA		NSGA-II		Reference Point
	Median (%)	Interquartile Range	Median (%)	Interquartile Range	(Time (s), Power (kW))
Gaussian	57.48	1.31	57.07	1.07	(1300, 113)
Gauss-Jordan	58.94	1.47	57.69	1.94	(3000, 130)
LU	56.85	0.88	53.19	1.23	(2000, 120)

We think that the better behaviour of MO-FA (regarding NSGA-II) could be due to the following: NSGA-II generates new solutions by applying a recombination methodology that considers only two parent solutions to obtain a child solution. MO-FA goes a step further and generates new solutions considering all the information gathered by the entire swarm (population). This collective behaviour could be the reason for the obtaining of better results.

On the other hand, a comparison between MO-FA and the real grid schedulers, WMS and DBC, is performed. WMS has been executed with the most efficient scheduling option, which consists in sorting the resources according to their response time. Results show that MO-FA always obtains better results for energy consumption (and also for the execution time). In fact, MO-FA reduces the

Table 3. Set coverage comparison of MO-FA and NSGA-II per each workflow

Coverage A ≥ B					
Algorithm		Workflows			Average
A	B	Gaussian	Gauss-Jordan	LU	
MO-FA	NSGA-II	60.00%	57.14 %	40.00%	52.38%
NSGA-II	MO-FA	16.66%	25.00 %	16.66%	19.44%

Table 4. Results of DBC, WMS and MO-FA per workflow. Time (s) and Power (kW)

Workflows	DBC		WMS		MO-FA	
	Time	Power	Time	Power	Time	Power
Gaussian	480.82	24.69	482.68	42.93	453.18	17.86
Gauss-Jordan	533.41	46.29	534.70	80.06	515.17	37.93
LU	596.66	34.91	612.29	58.55	586.29	26.78

Table 5. Study restricted by deadline to check the jobs executed successfully

Workflows	Constraint	DBC			WMS			MO-FA		
	Deadline	Time	Power/ Job	Jobs	Time	Power/ Job	Jobs	Time	Power/ Job	Jobs
Gaussian	460	460.43	2.31	10	460.00	4.01	10	453.18	1.48	12
	445	445.64	2.42	10	445.00	4.16	9	439.12	1.59	12
	430	430.73	2.40	9	430.00	4.16	9	415.82	2.33	12
Gauss-Jordan	530	525.71	2.99	15	530.00	5.34	14	515.17	2.16	15
	515	505.82	3.23	15	515.00	5.34	14	514.38	2.91	15
	500	500.08	3.08	14	500.00	5.34	14	496.27	3.06	15
LU	560	560.80	2.63	12	560.00	4.53	12	559.87	2.23	14
	545	545.00	2.81	12	545.00	4.90	10	537.70	2.40	14
	530	530.10	2.85	10	530.00	4.90	10	530.03	3.22	14

energy consumption in more than 50% respect to the results obtained by WMS and also a reduction around of 25% is performed regarding the results from DBC. Moreover, MO-FA obtains the minimum values for response time (Table 4). In the last comparison, we have evaluated a deadline restriction proving the decrease of successful jobs from DBC and WMS while MO-FA always executes successfully all the jobs with the minimum energy and time consumption (Table 5). Other advantage that MO-FA offers as a multi-objective algorithm is that it obtains more than one solution per each execution regarding DBC and WMS that only obtain one per each workflow.

5 Conclusions

Green computing is a hot topic nowadays being the Grid scheduling problem a challenging task to optimize the energy consumption. In this paper, not only an energy consumption optimization is considered but also the execution time of workflows with dependent jobs by using a novel multi-objective swarm approach.

MO-FA is inspired in the fireflies behaviour considering their brightness attraction as the minimization of energy consumption and execution time. Results demonstrate not just the goodness of MO-FA as a multi-objective algorithm but also its efficiency regarding real grid schedulers as WMS and DBC in all the cases. In future works, other multi-objective algorithms will be compared with MO-FA.

References

1. Arsuaga-Ríos, M., Vega-Rodríguez, M.A.: Multi-objective Firefly Algorithm for Energy Optimization in Grid Environments. In: Dorigo, M., Birattari, M., Blum, C., Christensen, A.L., Engelbrecht, A.P., Groß, R., Stützle, T. (eds.) ANTS 2012. LNCS, vol. 7461, pp. 350–351. Springer, Heidelberg (2012)
2. Bodenstein, C.: Heuristic scheduling in grid environments: Reducing the operational energy demand. In: Neumann, D., Baker, M., Altmann, J., Rana, O. (eds.) Economic Models and Algorithms for Distributed Systems. Autonomic Systems, pp. 239–256. Birkhäuser, Basel (2010)
3. Buyya, R., Murshed, M., Abramson, D.: A deadline and budget constrained cost-time optimisation algorithm for scheduling task farming applications on global grids. In: Int. Conf. on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA, pp. 2183–2189 (2002)
4. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multi-objective genetic algorithm: Nsga-ii. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
5. Diaz, C.O., Guzek, M., Pecero, J.E., Bouvry, P., Khan, S.U.: Scalable and energy-efficient scheduling techniques for large-scale systems. In: Proceedings of the 2011 IEEE 11th International Conference on Computer and Information Technology, CIT 2011, pp. 641–647. IEEE Computer Society, Washington, DC (2011)
6. Dolz, M.F., Fernández, J.C., Iserte, S., Mayo, R., Quintana, E.S., Cotallo, M.E., Díaz, G.: Energysaving cluster experience in ceta-ciemat. In: Ibergrid (ed.) 5th Iberian Grid Infrastructure Conference, Santander, Spain, pp. 39–50 (2011)
7. Entezari-Maleki, R., Movaghar, A.: A probabilistic task scheduling method for grid environments. Future Gener. Comput. Syst. 28(3), 513–524 (2012)
8. Foster, I., Kesselman, C.: The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers Inc., San Francisco (2003)
9. Hernández, C.J.B., Sierra, D.A., Varrette, S., Pacheco, D.L.: Energy efficiency on scalable computing architectures. In: Proceedings of the 2011 IEEE 11th International Conference on Computer and Information Technology, CIT 2011, pp. 635–640. IEEE Computer Society, Washington, DC (2011)
10. Khan, S.U., Ahmad, I.: A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Trans. Parallel Distrib. Syst. 20(3), 346–360 (2009)
11. Khan, S.U.: A goal programming approach for the joint optimization of energy consumption and response time in computational grids. In: 28th IEEE International Performance Computing and Communications Conference, pp. 410–417 (2009)
12. Khan, S.U.: A multi-objective programming approach for resource allocation in data centers. In: The 2009 International Conference on Parallel and Distributed Processing Techniques and Applications, pp. 152–158 (2009)

13. Khare, V., Yao, X., Deb, K.: Performance Scaling of Multi-objective Evolutionary Algorithms. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 376–390. Springer, Heidelberg (2003)
14. Lee, Y.C., Zomaya, A.Y.: Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans. Parallel Distrib. Syst. 22(8), 1374–1381 (2011)
15. Lee, Y.H., Leu, S., Chang, R.S.: Improving job scheduling algorithms in a grid environment. Future Gener. Comput. Syst. 27(8), 991–998 (2011)
16. Lindberg, P., Leingang, J., Lysaker, D., Khan, S.U., Li, J.: Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. The Journal of Supercomputing 59(1), 323–360 (2012)
17. Liu, W., Li, H., Du, W., Shi, F.: Energy-aware task clustering scheduling algorithm for heterogeneous clusters. In: Proceedings of the 2011 IEEE/ACM International Conference on Green Computing and Communications, GREENCOM 2011, pp. 34–37. IEEE Computer Society, Washington, DC (2011)
18. Lovasz, G., Berl, A., De Meer, H.: Energy-efficient and performance-conserving resource allocation in data centers. In: Proc. of the COST Action IC0804 on Energy Efficiency in Large Scale Distributed Systems - 2nd Year, pp. 31–35. IRIT (2011)
19. Sulistio, A., Poduval, G., Buyya, R., Tham, C.: On incorporating differentiated levels of network service into gridsim. Future Gener. Comput. Syst. 23(4), 606–615 (2007)
20. Talukder, A.K.M.K.A., Kirley, M., Buyya, R.: Multiobjective differential evolution for scheduling workflow applications on global grids. Concurr. Comput.: Pract. Exper. 21(13), 1742–1756 (2009)
21. Tsai, M.-Y., Chiang, P.-F., Chang, Y.-J., Wang, W.-J.: Heuristic Scheduling Strategies for Linear-Dependent and Independent Jobs on Heterogeneous Grids. In: Kim, T.-h., Adeli, H., Cho, H.-s., Gervasi, O., Yau, S.S., Kang, B.-H., Villalba, J.G. (eds.) GDC 2011. CCIS, vol. 261, pp. 496–505. Springer, Heidelberg (2011)
22. Tsuchiya, T., Osada, T., Kikuno, T.: Genetics-based multiprocessor scheduling using task duplication. Microprocessors and Microsystems 22(3-4), 197–207 (1998)
23. Wang, L., von Laszewski, G., Dayal, J., Wang, F.: Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with dvfs. In: Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, CCGRID 2010, pp. 368–377. IEEE Computer Society, Washington, DC (2010)
24. Yang, X.-S.: Firefly Algorithms for Multimodal Optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009)
25. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 292–304. Springer, Heidelberg (1998)

A Framework for Derivative Free Algorithm Hybridization

Jose Luis Espinosa-Aranda, Ricardo Garcia-Rodenas, and Eusebio Angulo

Universidad de Castilla-La-Mancha, Paseo de la Universidad 4, Ciudad Real, Spain
{JoseL.Espinosa,Ricardo.Garcia,Eusebio.Angulo}@uclm.es

Abstract. Column generation is a basic tool for the solution of large-scale mathematical programming problems. We present a class of column generation algorithms in which the columns are generated by derivative free algorithms, like population-based algorithms. This class can be viewed as a framework to define hybridization of free derivative algorithms. This framework has been illustrated in this article using the Simulated Annealing (SA) and Particle Swarm Optimization (PSO) algorithms, combining them with the Nelder-Mead (NM) method. Finally a set of computational experiments has been carried out to illustrate the potential of this framework.

Keywords: Column generation, derivative free methods, Particle Swarm Optimization, Nelder-Mead Simplex, Simulated Annealing.

1 Introduction

Derivative free algorithms is a key area with an increasing interest due to its versatility and broad use in optimization problems. The main motivation for this paper is to propose a means to accelerate the convergence of these algorithms in order to be able to apply them to large-scale problems in a reasonable computational time.

Column generation (CG) [1] is a classic means to attack the following (large-scale) constrained optimization problem:

$$\underset{x \in X}{\text{minimize}}\ f(x), \qquad \mathrm{P}(f, X)$$

where the feasible region $X \subseteq \mathbb{R}^N$ is non-empty and closed, and $f : X \mapsto \mathbb{R}$ is a continuous function on X. These algorithms follow two main steps:

1. Column Generation Problem (CGP). A relaxation of the original problem is constructed, based on current estimates $\widetilde{x}$ of the optimal solution, which provides a bound to the optimal value of the original problem, a new column and information that indicates if the column should be introduced into the solution or if the process should stop. This problem can be defined by $\mathrm{P}(\hat{f}, X)$ where $\hat{f}(x)$ is an approximation of $f(x)$ on $\tilde{x}$.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 80–89, 2013.

2. Restricted Master Problem (RMP). Previously generated columns define an inner approximation $\widehat{X}$ of the feasible region X and the new column $\widetilde{y}$ is used to expand it. The original problem is approximated by $\mathrm{P}(f, \widehat{X})$ and its solution $\widetilde{x}$ is used to define a new CG sub-problem which iterates the process.

Classic examples of this type of algorithm used for linearly and nonlinearly constrained problems are *simplicial decomposition* (SD) [2] and *Restricted Simplicial Decomposition* [3]. In these algorithms the relaxation is a linearization of the objective function while the feasible set is defined by the convex hull of the retained columns in the RMP. In this algorithm column generation is not based on pricing or dual information and is associated with multidimensional extensions of primal descent algorithms in NLP.

In [1] a CG algorithm is proposed based on closed descent algorithms. In this framework the CGP is defined by a given algorithm and the RMP as a phase which tries to accelerate the algorithm used in CGP. RMP is solved by an algorithm capable of using the advantages of RMP to its favour, like a reduced number of variables and a simple set of restrictions.

In this algorithm the SD is obtained as a column running only one iteration of the *Frank-Wolfe* method [4]. A key theoretical result is that the local rate of convergence of SD is governed by the local convergence rate of the method chosen for the solution of the RMP; thus a superlinear or quadratic convergence rate may be attained if a (projected) Newton method is used [5]. Note that the rate of convergence of the *Frank-Wolfe* algorithm is sublinear.

In [6] was carried out an experimental study focused on their computational efficiency of CG methods. Two types of test problems were considered, the first one is the nonlinear, capacitated single-commodity network flow problem, and the second one is a combined traffic assignment model. This paper validates this methodology in order to improve the performance of feasible direction and simplicial decomposition methods used in equilibrium assignment models.

The main contribution of this paper is to extend this framework to derivative free optimization methods for general optimization problems. To verify the goodness of the framework a set of computational tests have been performed with the algorithms: i) Nelder-Mead simplex method (NM) [7] ii) Bohachevsky et al. Simulated Annealing (SA) [8], and iii) Particle Swarm Optimization (PSO) [9], showing that the convergence rate and effectiveness of the heuristics algorithms can be improved by hybridization.

The paper is organized as follows. In Section 2 the proposed framework for derivative free optimization methods is explained, in Section 3 we discuss some instances of the hybrid CG algorithm, in Section 4 several computational experiments are reported, and finally in Section 5 we conclude with a discussion of our findings and future work.

2 The Conceptual Framework of Hybridization

[1] dealt with a class of CG algorithms in which the approximated objective function $\hat{f}(y)$ coincides with the original $f(y)$. This point of view leads to a CG algorithm which can be defined according to the following three key items: i) the algorithm to solve the $\mathrm{P}(f, X)$ (denoted by $\mathcal{A}_{\mathrm{c}}$), ii) the algorithm applied to RMP (denoted by $\mathcal{A}_{\mathrm{r}}$) and iii) the means of stating the inner approximation $\widehat{X}$. The algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$ analysed in [1] are descent primal methods and the $\mathrm{P}(f, X)$ is a differentiable optimization problem. In this paper we extend this framework to derivative free optimization methods and for general optimization problems $\mathrm{P}(f, X)$. We begin by stating the characteristics of the optimization algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$ that can be used to solve the $\mathrm{CGP}(f, X)$ or $\mathrm{RMP}(f, \widehat{X})$.

A type of derivative free algorithm which works with populations generates an evolution of the population instead of generating a sequence of solutions (*particles* in Particle Swarm Optimization, an *atom* in Simulated Annealing, *chromosomes* in Genetic Algorithms, *ants* in Ant Colony Optimization, etc. [10]). For this reason we consider algorithms based on populations, and if the cardinality of this population is one the classical optimization methods appear.

Assumption 1 (Optimization Algorithm) *Let $P(f, Z)$ be an optimization problem and let $\mathcal{A}$ be an optimization algorithm. This algorithm is defined as an iterative procedure which will be assumed to fulfill the following two conditions.*

i) (Feasible population). This algorithm works on a population of particles $\widetilde{\mathcal{Z}} = \{z_1, \cdots, z_m\}$ which is modified iteratively. This algorithm is described by means of a point-to-set algorithmic mapping

$$\begin{aligned} \mathcal{A} : Z^m &\mapsto 2^{Z^m} \\ \widetilde{\mathcal{Z}} &\mapsto \mathcal{A}(\widetilde{\mathcal{Z}}) \end{aligned} \tag{1}$$

where 2^{Z^m} is the power set of Z^m. Also we denote

$$\mathcal{A}^t(\widetilde{\mathcal{Z}}) := [\mathcal{A} \overbrace{\circ \cdots \circ}^{t \text{ times}} \mathcal{A}]\,(\widetilde{\mathcal{Z}}) \tag{2}$$

and the realization of $t'-$iterations generate a sequence of populations $\widetilde{\mathcal{Z}}^1, \cdots, \widetilde{\mathcal{Z}}^{t'}$ such as $\widetilde{\mathcal{Z}}^t := \{z_1^t, \cdots, z_m^t\} \in \mathcal{A}(\mathcal{Z}^{t-1})$ of feasible points for any initialization $\widetilde{\mathcal{Z}} \in Z^m$.

ii) (Convergent property). This algorithm is convergent for every $\widetilde{\mathcal{Z}} \in Z^m$ in the following sense. Let $\mathcal{Z}^t \in \mathcal{A}^t(\mathcal{Z})$, $t = 1, 2, \cdots$ be the sequence of populations generated by $\mathcal{A}$ and let

$$z^t := \operatorname*{Arg\,minimize}_{z \in \mathcal{Z}^t} f(z) \tag{3}$$

then

$$d_{SOL(f,Z)}(z^t) := \left\{ \min_{z^* \in SOL(f,Z)} \|z^t - z^*\| \right\} \to 0 \tag{4}$$

where $SOL(f, Z)$ denotes the set of global minimizers of $P(f, Z)$.

The methods used in [1] are descent closed algorithms which are convergent for differentiable convex programs. In this paper we generalize them to convergent algorithms for a given optimization problem. Properties i) and ii) guarantee the convergence of the algorithm in a finite number of iterations or a finite number of descents in the objective function. In the CG method the number of these descents is used to decide a change between the algorithms used. The following definition explains this concept.

Definition 1 (*n−descent iteration).*
We say that an algorithm $\mathcal{A}$ achieves an $n-$descent if the algorithm is applied in a sufficient number of iterations to ensure n times a descent in the objective function. More formally, let n be a positive integer number and we denote

$$f(\widetilde{\mathcal{Z}}) := \underset{z \in \widetilde{\mathcal{Z}}}{\text{minimize}}\, f(z) \tag{5}$$

An $n-$ descent iteration of $\mathcal{A}$ consists of applying the following algorithm:

$$\begin{cases} \text{Let } \widetilde{\mathcal{Z}}' \text{ be the initial population and let } \ell = 0. \\ \text{Do While } (\ell \leq n) \\ \quad \widetilde{\mathcal{Z}} \in \mathcal{A}(\widetilde{\mathcal{Z}}') \\ \quad \text{If } f(\widetilde{\mathcal{Z}}') < f^* \text{then } f^* = f(\widetilde{\mathcal{Z}}') \text{ and } \ell = \ell + 1 \\ \quad \widetilde{\mathcal{Z}}' = \widetilde{\mathcal{Z}} \\ \text{End Do While} \end{cases}$$

The realization of a $n-$descent iteration is denoted by:

$$\widetilde{\mathcal{Z}} \in \mathcal{A}(\widetilde{\mathcal{Z}}', n) \tag{6}$$

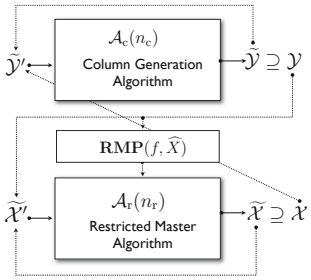


Fig. 1. Hybridization of Algorithms

In Table 1, we summarize the different steps of a CG algorithm belonging to the proposed framework. The resulting algorithm can be viewed as a hybrid algorithm of $\mathcal{A}_c$ and $\mathcal{A}_r$.

Table 1. Hybrid CG algorithm

0. (*Initialization*): Let $\{n_{\mathrm{c}}^t\}$ and $\{n_{\mathrm{r}}^t\}$ be two sequences of positive integer numbers. Let $\widetilde{\mathcal{Y}}'^1$ and $\widetilde{\mathcal{X}}'^1$ be respectively the initial populations for the algorithms $\mathcal{A}_{\mathrm{c}}$ and $\mathcal{A}_{\mathrm{r}}$. Let $\mathcal{X}^1 := \{\emptyset\}$ and $t := 1$.
1. (*Column Generation Algorithm*): Apply a n_{c}^t -descent with the algorithm $\mathcal{A}_{\mathrm{c}}$ on the problem $\mathrm{P}(f, X)$, starting from $\widetilde{\mathcal{Y}}'^t$. Let $\widetilde{\mathcal{Y}}^t$ be the resulting population i.e., $\widetilde{\mathcal{Y}}^t \in \mathcal{A}(\widetilde{\mathcal{Y}'}^t, n_{\mathrm{c}}^t)$ and let y^t be the current best solution. i.e.,

$$y^t := \underset{y \in \widetilde{\mathcal{Y}}^t}{\text{Arg minimize}}\ f(y), \tag{7}$$

2. (*Set augmentation*): Choose a set of columns $\mathcal{Y}^t$ satisfying $y^t \in \mathcal{Y}^t \subset \widetilde{\mathcal{Y}}^t$. Let $X^{t+1} \subset X$ a nonempty closed set containing $\{\mathcal{Y}^t, \mathcal{X}^t\}$.
3. (*Update of populations for RMP*$(f, \widehat{X})$): Let $\widetilde{\mathcal{X}'}^{t+1} \subseteq \mathcal{Y}^t \cup \widetilde{\mathcal{X}}^t$.
4. (*Restricted Master Algorithm*): Apply a n_{r}^t-descent with the algorithm $\mathcal{A}_{\mathrm{r}}$ on $\mathrm{RMP}(f, X^{t+1})$, starting from $\widetilde{\mathcal{X}'}^{t+1}$. Let $\widetilde{\mathcal{X}}^{t+1} \in \mathcal{A}(\widetilde{\mathcal{X}'}^{t+1}, n_{\mathrm{r}}^t)$ be the resulting population. Let x^{t+1} be the current best solution. i.e.,

$$x^{t+1} := \underset{x \in \widetilde{\mathcal{X}}^{t+1}}{\text{Arg minimize}}\ f(x), \tag{8}$$

5. (*Update of population for* $P(f, X)$): Choose a set of solutions $\mathcal{X}^{t+1}$ satisfying $x^{t+1} \in \mathcal{X}^{t+1} \subseteq \widetilde{\mathcal{X}}^{t+1}$ and update the population for solving $\mathrm{P}(f, X)$ as $\widetilde{\mathcal{Y}'}^{t+1} \subseteq \widetilde{\mathcal{Y}}^t \cup \mathcal{X}^{t+1}$.
6. (*Termination criterion*): If $x^{t+1} \in \mathrm{SOL}(f, X) \rightarrow$ Stop. Otherwise, let $t := t + 1$. Go to Step 1.

3 Instances of Hybrid CG Algorithms

Classical algorithms used in differentiable optimization such as SD, RSD or NSD [1] are roughly obtained letting Z be a point instead of a population, X a polyhedral set and $\widehat{X}$ as the convex hull of retained columns. These methods can be interpreted as a hybridization of $\mathcal{A}_{\mathrm{c}}$=Frank-Wolfe and $\mathcal{A}_{\mathrm{r}}$=(projected) Newton method.

In free derivatives optimization the line-search based modification of the Hooke and Jeeves method [11] combine, in one iteration, a coordinate-wide search through each of the variables ($\mathcal{A}_{\mathrm{c}}$) with a pattern search ($\mathcal{A}_{\mathrm{r}}$). The $\mathcal{A}_{\mathrm{r}}$ produces an acceleration in the convergence of $\mathcal{A}_{\mathrm{c}}$.

In this paper we investigate numerically only the basic framework which consists of letting $X = \mathbb{R}^n$, $\widehat{X}^t = \mathbb{R}^N$, $\mathcal{Y}^t = \{y^t\}$, $\widetilde{\mathcal{X}}^{t+1} = \{y^t\}$ and $\widetilde{\mathcal{Y}'}^t = \widetilde{\mathcal{Y}}^{t-1}$. Roughly, this basic hybrid algorithm consists of interchanging both algorithms when an $n-$descent iteration is carried out. We have introduced the following improvement. In order to take into account the random walking of $\mathcal{A}_{\mathrm{c}}$, the

objective function $f(x)$ in RMP is represented by $f(d^t \cdot x^T)$ where T is the transpose of a vector, and $d^t = y^t - x^{t-1}$. This new function is a scalarization of the function $f(x)$.

In the numerical experiments, for $\mathcal{A}_r$ the Nelder-Mead simplex search method has been used. This method is a direct search method that does not use numerical or analytic gradients and has local convergence. This algorithm described in [12] uses a simplex of $N+1$ points for N-dimensional vectors x. The algorithm first makes a simplex around the initial point y^t. Then, the algorithm modifies the simplex repeatedly generating at each step of the search a new point in or near the current simplex. The function value at the new point is compared with the function's values at the vertices of the simplex and, usually, one of the vertices is replaced by the new point, creating a new simplex.

On the other hand, for $\mathcal{A}_c$ two algorithms with properties of global convergence have to be employed. The motivation is to combine global convergent methods with a local convergent method which is more computationally efficient. In particular in this paper SA and PSO algorithms have been used for $\mathcal{A}_c$.

The SA algorithm used in this paper was proposed by [8] is an improved version of the classical SA [13]. It adds the parameters α and β, α is the maximum step size and β dictates the conditional probability that a worse solution can be accepted.

The second algorithm used is the original PSO algorithm using the global *gbest* model. It is a population-based algorithm and evolutionary in nature, introduced by [9]. PSO is a kind of random search algorithm based on the metaphors of social interaction and communications. This method has been shown to be effective in solving difficult and complex optimization problems in a wide range of fields [14] [15]. PSO maintains at each iteration a set of swarm particles, feasible points represented by $(\widetilde{\mathcal{Y}}')$ in our framework.

4 Computational Results

In this section the framework proposed in this paper is tested by the hybridization of SA with NM (SA+NM) and PSO with NM (PSO+NM) to test the improvement of the algorithms used. As mentioned above, the main motivation of this hybridization is to combine the capacity of the SA and PSO algorithms to find a global minimum with the capacity of the NM algorithm to obtain a more precise solution near a local minimum.

In the next sections, the design of experiments is explained, and the results are reported comparing our approach with the SA, NM and PSO algorithms.

4.1 Design of the Experiments

To compare the quality of the proposed modified algorithms it is necessary to test them with a large-scale test process on a variety of response functions, ignoring the possible effectiveness of the algorithm in a specific type of function.

The set of 20 deterministic functions used in the computational experiments are found in [16] and [17], and have been selected because they are a set of curvilinear functions for difficult unconstrained optimization problems with a variety of dimensions (N) and functional forms, which makes it possible to asses the robustness of the proposed approach.

To evaluate the algorithms' effectiveness each of them have been executed 100 times per test function. In each of the executions, the initial point in SA, NM and SA+NM is sampled from Uniform(-50,50) and initial particles in PSO and PSO+NM are randomly generated from Uniform(-50,50). Also, in PSO and PSO+NM algorithms the number of particles is defined as $5N$.

The stopping criterion selected for all the algorithms is that the current execution reaches $5000N^2$ function evaluations, which corresponds to $1000N$ iterations of the PSO algorithm. The sequence of number of descents was $\{n_{\mathrm{c}}^t\} = 5$ and $\{n_{\mathrm{r}}^t\} = 100$ in the numerical tests.

The criteria selected to compare the algorithm's robustness, effectiveness, efficiency and accuracy using the results obtained in the experiments described above are:

1. Rate of successful minimizations, considering that a successful minimization occurs when $|f(\mathcal{X}^t) - f^*| < 10^{-10}$ where $f^* = \min\{f(x) : x \in X\}$.
2. Average of function evaluations until a successful minimization occurs.
3. Gap between the best minimum found in the 100 executions and the optimum of the function.

4.2 Results

The results given in this section will show that the approach described is capable of improving the SA, NM and PSO algorithms.

Tables 2 and 3 show the complete results of the experiments performed. By comparing them, it can be seen that the modified version of the algorithms in general has a significantly higher rate of successful minimization and a lower average of objective function evaluation before reaching a successful minimization than the original algorithms.

To prove these statements the non-parametric statistical Wilcoxon test is carried out with a significance of 0.05 to compare the paired groups SA vs SA+NM, NM vs SA+NM, PSO vs PSO+NM and NM vs PSO+NM, and gives the results shown in Table 4. Taking into account these results, it can be stated that there exists a clear improvement in the number of successful minimizations for all the algorithms.

In the case of function evaluations until a successful minimization occurs, the results show that in NM and SA vs SA+NM there are no conclusive results because of the lack of successful minimizations in the NM and SA algorithms, but in PSO and NM vs PSO+NM it can be seen that there does exist an improvement, reducing the number of evaluations necessary to find the minimum of the function.

Table 2. NM, SA and SA+NM Results

	Successful minimizations			Function Evaluations			Gap with best minimum found		
Function	NM	SA	SA+NM	NM	SA	SA+NM	NM	SA	SA+NM
1	17	0	17	1.829e+02	-	1.479e+04	1.769e-13	2.190e-08	2.827e-15
2	0	0	24	-	-	1.816e+04	1.411e-09	1.035e-05	3.919e-12
3	5	0	46	2.912e+02	-	1.450e+04	4.300e-11	1.191e-06	1.943e-13
4	0	0	97	-	-	1.658e+04	3.756e-10	7.366e-07	9.437e-16
5	0	0	2	-	-	4.504e+04	2.181e-10	1.885e-04	4.288e-11
6	0	0	87	-	-	4.480e+04	3.941e-10	6.979e-06	2.553e-13
7	69	5	64	4.014e+02	2.757e+04	2.210e+04	0	2.208e-12	1.449e-16
8	0	0	0	-	-	-	3.169e-10	3.652e-04	1.167e-07
9	0	0	29	-	-	7.993e+04	4.480e-10	8.617e-05	2.130e-12
10	0	0	0	-	-	-	1.261e-10	1.809e-03	5.536e-09
11	0	0	23	-	-	8.001e+04	9.319e-10	6.986e-05	3.390e-13
12	0	0	0	-	-	-	6.552e-08	3.071e-06	1.173e-06
13	0	0	0	-	-	-	5.403e-07	3.247e-06	1.175e-06
14	0	0	29	-	-	3.200e+05	7.027e-10	1.916e-03	6.095e-12
15	78	0	0	3.499e+03	-	-	1.704e-14	6.399e-03	4.059e-06
16	0	0	0	-	-	-	1.009e-01	6.264e-01	5.793e-01
17	0	0	0	-	-	-	235.796	5.081	1.990e+00
18	0	0	0	-	-	-	2.949e-01	1.589	1.947e+00
19	0	0	0	-	-	-	2.544	1.406	1.686e+00
20	0	0	0	-	-	-	7.227e-01	4.031	4.348e+00

Table 3. NM, PSO and PSO+NM Results

	Successful minimizations			Function Evaluations			Gap with best minimum found		
Function	NM	PSO	PSO+NM	NM	PSO	PSO+NM	NM	PSO	PSO+NM
1	17	48	64	1.829e+02	5.408e+03	6.862e+02	1.769e-13	0	0
2	0	64	76	-	2.962e+03	2.159e+03	1.411e-09	0	0
3	5	58	69	2.912e+02	2.778e+03	9.795e+02	4.300e-11	0	0
4	0	57	71	-	3.023e+03	1.677e+03	3.756e-10	0	0
5	0	61	80	-	1.112e+04	5.231e+03	2.181e-10	1.523e-109	3.641e-99
6	0	62	80	-	3.925e+03	1.956e+03	3.941e-10	0	6.043e-283
7	69	57	96	4.014e+02	7.442e+03	1.307e+03	0	0	1.733e-238
8	0	0	63	-	-	8.959e+03	3.169e-10	1.675e-09	3.748e-23
9	0	24	34	-	9.891e+03	4.012e+03	4.480e-10	1.868e-32	1.805e-31
10	0	25	64	-	5.543e+04	2.197e+04	1.261e-10	7.500e-22	0
11	0	57	73	-	9.798e+03	6.319e+03	9.319e-10	0	0
12	0	43	43	-	1.284e+05	8.431e+04	6.552e-08	6.626e-12	6.626e-12
13	0	1	23	-	3.151e+05	2.354e+05	5.403e-07	9.539e-11	7.986e-11
14	0	30	37	-	4.586e+04	2.528e+04	7.027e-10	1.032e-30	5.311e-31
15	78	0	43	3.499e+03	-	8.342e+04	1.704e-14	1.728e-09	5.276e-18
16	0	0	0	-	-	-	1.009e-01	7.411e-03	1.478e-02
17	0	13	10	-	2.495e+05	2.193e+05	2.358e+02	0	0
18	0	0	3	-	-	3.383e+05	2.949e-01	1.865e-08	5.417e-18
19	0	44	53	-	5.563e+05	4.949e+05	2.544e+00	0	0
20	0	36	22	-	1.371e+06	8.114e+05	7.227e-01	0	0

Table 4. Significances of Wilcoxon tests

Algorithms	S. minimizations	F. Evaluations	Gap best minimum
NM vs SA+NM	0.037	0.0545*	0.3005
SA vs SA+NM	0.0025	-*	0.0285
NM vs PSO+NM	0.00015	0.034	0.0001
PSO vs PSO+NM	0.001	0.0001	0.2345

* Not enough data

To illustrate the general performance of the approaches in terms of evaluations of the objective function, Figure 2 show the 2nd function of the experiments, (B2 function), plotting the best evaluation of the function vs the number of evaluations of the objective function. It can be seen from the figures that the SA+NM and PSO+NM algorithms converge more quickly than the classical algorithms, also breaking the local minimum which cannot be reduced by the NM algorithm.

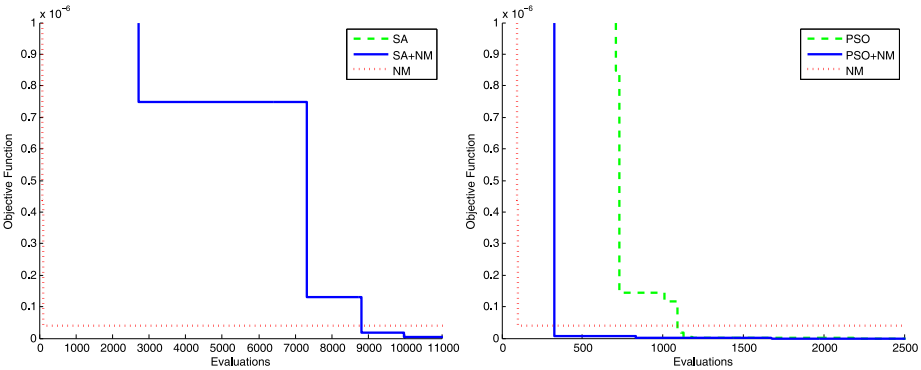


Fig. 2. Objective Function *vs* Evaluations for SA, NM and SA+NM

Finally, for the gap with the best found minimum results it can be shown that the SA is improved by the SA+NM algorithm and the NM is improved by the PSO+NM algorithm.

5 Conclusions

In this paper a new framework for hybridization of derivative free algorithms is presented. This approach includes the hybridization of two methods in an attempt to combine the specific capacity of exploitation and exploration of each algorithm, increasing the robustness, effectiveness, efficiency and accuracy.

This paper investigates numerically the basic CG algorithm consisting of letting $\widehat{X} = X = \mathbb{R}^N$ and the algorithms NM, SA and PSO. Also a set of computational tests has been done using 20 curvilinear functions for difficult unconstrained optimization, in which the original algorithms are compared with the modified versions. The results show that the hybrid algorithms have a higher rate of successful minimizations and an acceleration of the algorithms, which are capable of giving the optimum using fewer evaluations of the function than the original algorithm.

Future work will research new instances of the algorithm based on the initialization [16] or augmentation rules for defining $\widehat{X}$. The idea is to extend the computational results by trying to apply the framework to other algorithms and more complex optimization problems like [18].

Acknowledgements. This research has been financed by the *Ministerio de Ciencia e Innovación* of Spain with the TRA-2011-27791-C03-03 research project.

References

1. García, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, I: Convergence analysis. Optimization 52(2), 171–200 (2003)
2. Von Hohenbalken, B.: Simplicial decomposition in nonlinear programming algorithms. Mathematical Programming 13(1), 49–68 (1977)
3. Ventura, J.A., Hearn, D.W.: Restricted simplicial decomposition for convex constrained problems. Mathematical Programming 59, 71–85 (1993)
4. Frank, M., Wolfe, P.: An algorithm for quadratic programming. Naval Research Logistics Quarterly, 95–110 (1956)
5. Hearn, D.W., Lawphongpanich, S., Ventura, J.A.: Restricted simplicial decomposition: Computation and extensions. Mathematical Programming Study 31, 99–118 (1987)
6. García-Ródenas, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, II: Numerical investigations. Computers and Operations Research 38(3), 591–604 (2011)
7. Nelder, J.A., Mead, R.: A simplex method for function minimization. Computer Journal 7, 308–313 (1965)
8. Bohachevsky, I.O., Johnson, M.E., Myron, L.S.: Generalized Simulated Annealing for Function Optimization. Technometrics 28(3) (1986)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings IEEE of the International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
10. Kennedy, J.F., Kennedy, J., Eberhart, R.C., Shi, Y.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)
11. Hooke, R., Jeeves, T.A.: Direct search solution of numerical and statistical problems. Journal of the Association for Computing Machinery, 212–229 (1961)
12. Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the nelder-mead simplex method in low dimensions. SIAM Journal on Optimization 9(1), 112–147 (1999)
13. Laarhoven, P., Aarts, E.: Simulated annealing: Theory and Applications, 3rd edn. Kluwer Academic Publishers, Dordrecht (1987)
14. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. Journal of Artificial Evolution and Applications, 1–10 (2008)
15. Angulo, E., Castillo, E., García-Ródenas, R., Sánchez-Vizcaíno, J.: Determining Highway Corridors. Journal of Transportation Engineering 138(5), 557–570 (2012)
16. Fan, S.-K.S., Zahara, E.: A hybrid simplex search and particle swarm optimization for unconstrained optimization. European Journal of Operational Research, 527–548 (2006)
17. Functions definition, `http://bit.ly/UQlMTB` (last access: January 15th, 2013)
18. Espinosa-Aranda, J.L., García-Ródenas, R.: A discrete event-based simulation model for real-time traffic management in railways. Journal of Intelligent Transportation Systems 16(2), 94–107 (2012)

PSO-Tagger: A New Biologically Inspired Approach to the Part-of-Speech Tagging Problem

Ana Paula Silva[1], Arlindo Silva[1], and Irene Rodrigues[2]

[1] Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco
{dorian,arlindo}@ipcb.pt
[2] Universidade de Évora
ipr@uevora.pt

Abstract. In this paper we present an approach to the part-of-speech tagging problem based on particle swarm optimization. The part-of-speech tagging is a key input feature for several other natural language processing tasks, like phrase chunking and named entity recognition. A tagger is a system that should receive a text, made of sentences, and, as output, should return the same text, but with each of its words associated with the correct part-of-speech tag. The task is not straightforward, since a large percentage of words have more than one possible part-of-speech tag, and the right choice is determined by the part-of-speech tags of the surrounding words, which can also have more than one possible tag. In this work we investigate the possibility of using a particle swarm optimization algorithm to solve the part-of-speech tagging problem supported by a set of disambiguation rules. The results we obtained on two different corpora are amongst the best ones published for those corpora.

Keywords: Part-of-speech Tagging, Disambiguation Rules, Evolutionary Algorithms, Particle Swarm Optimization, Natural Language Processing.

1 Introduction

The words in most languages can assume different roles in a sentence, depending on how they are used. These roles are normally designated by part-of-speech (POS) tags or word classes, such as nouns, verbs, adjectives and adverbs. The process of classifying words into their POS, and labeling them accordingly, is known as POS tagging, or, simply, tagging. Tagging is a very important task in natural language processing (NLP), because it is a necessary step in a large number of more complex processes like phrase chunking, named entity recognition, parsing, machine translation, information retrieval, speech recognition, etc. In fact, it is the second step in the typical NLP pipeline, following tokenization.

The role of a word in a sentence is determined by its surrounding words (context). For instance, the word *fish* can assume the function of a **verb**, "Men like to fish.", or a **noun**, "I like smoked fish", depending on how we choose to use it on a sentence. This means that in order to assign to each word of a

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 90–99, 2013.

sentence its correct tag, we have to consider the context in which each word appears. However, each of the words belonging to a word's context can also be used in different ways, and that means that in order to solve the problem we have to have some type of disambiguation mechanism. Traditionally there are two groups of methods used to tackle this task, with respect to the information model used. The first group is based on statistical data concerning the different context possibilities for a word (stochastic taggers) [1–4], while the second group is based on rules that capture the language properties and are used to improve tagging accuracy [5–7].

The simplest stochastic tagger, called unigram tagger, takes only into account the word itself. It assigns the tag that is most likely for one particular token. The tagger works like a simple lookup tagger, assigning to each word the most common tag for that word in the training corpus. To do that, the learning process just counts, for each word, the number of times it appears with each of the possible tags. A n-gram tagger is a generalization of an unigram tagger, whose context is the current word together with the part-of-speech tags of the n-1 preceding tokens. In this case, the training step saves, for each possible tag, the number of times it appears in every different context presented on the training corpus. Since the surrounding words can also have various possibilities of classification, it is necessary to use a statistical model that allows the selection of the best choices for marking the entire sequence, according to the model. Most of the stochastic taggers are based on hidden Markov models, and, because of that, a word's context consists only in the tags of the words that precede it.

One of the most popular taggers based on rules is the one proposed by Brill [5]. Brill's rules are usually called transformation rules. The system can be divided into two main components: a list of transformation rules patterns for error correction, and a learning system. The transformation patterns are hand made and provided to the learning algorithm, which will instantiate and order them. The search is made in a greedy fashion. The result is an ordered set of transformation rules, which is then used to perform the tagging. These rules are meant to correct mistakes in a pre-tagged text, usually achieved by a baseline system that marks each word with its most common tag. They are applied in a iterative way until no rule can be fired.

As already observed, the only information a n-gram tagger considers from prior context is the tags, even though words themselves might be an useful source of information. It is simply impractical for n-gram models to be conditioned by the context words themselves (and not only their tags). On the other hand, Brill's approach allows the inclusion of other type of information besides the context. In fact, the author also used the learning algorithm to achieve a set of lexicalized transformation rules, that includes not only the tags but the words themselves.

There are also some other aspects that can be used to determine a word's category beside its context in a sentence [8]. The internal structure of a word may give useful clues as to the word's class. For example, *-ness* is a suffix that combines with an adjective to produce a noun, e.g., *happy* → *happiness*, *ill* →

illness. Therefore, if we encounter a word that ends in *-ness*, it is very likely to be a noun. Similarly, *-ing* is a suffix that is most commonly associated with gerunds, like *walking, talking, thinking, listening*. We also might guess that any word ending in *-ed* is the past participle of a verb, and any word ending with *'s* is a possessive noun.

More recently, several evolutionary approaches have been proposed to solve the tagging problem. These approaches can also be divided by the type of information used to solve the problem, statistical information [2–4], and rule-based information [6]. Shortly, in the former, an evolutionary algorithm is used to assign the most likely tag to each word of a sentence, based on a training table, that basically has the same information that is used in the traditional probabilistic approaches. Notwithstanding, there is an important difference related with the context's shape, i.e they also take into account context information about the tags that follow a particular word. On the other hand, the later is inspired by the Brill's tagger. In this case a genetic algorithm (GA) is used to evolve a set of transformations rules, that will be used to tag a text in much the same way as the Brill's tagger. While in Araujo's work, the evolutionary algorithm is used to discover the best sequence of tags for the words of a sentence, using an information model based on statistical data, in Wilson's work the evolutionary algorithm is used to evolve the information model itself, in the form of a set of transformation rules.

In this work we investigate the possibility of using a discrete particle swarm optimization (PSO) algorithm to solve the POS tagging problem, using as information model a set of disambiguation rules extracted earlier by other evolutionary algorithm, which in this case was also a PSO based algorithm. The rules were extracted from an annotated corpus and have the typical form of a classification rule. The problem of searching for the best tag assignments can be seen as a combinatorial optimization problem, where a solution is evaluated with the help of the disambiguation rules previously learned. We decided to test the application of swarm intelligence to this problem, since other population based algorithms, in particular genetic algorithms, have been successfully applied to many combinatorial optimization tasks.

2 Particle Swarm Optimization

In PSO algorithms, a particle decides where to move next, considering its own experience, which is the memory of its best past position, and the experience of its most successful neighbor. There may be different concepts and values for neighborhood; it can be seen as spatial neighborhood where it is determined by the Euclidean distance between the positions of two particles, or as a sociometric neighborhood (e.g.: the index position in the storing array). The number of neighbors k usually considered is either $k = 2$ or $k = all$. Although some actions differ from one variant of PSO to the other, the common pseudo-code for PSO is as follows:

Algorithm 1. Generic Particle Swarm Optimization Algorithm

```
Initiate_Swarm()
repeat
  for p= 1 to number of particles do
    Evaluate(p)
    Update_past_experience(p)
    Update_neighborhood_best(p,k)
    for d= 1 to number of Dimensions do
      Move(p,d)
    end for
  end for
until Criterion
```

The output of this algorithm is the best point in the hyperspace the swarm visited. Since its introduction, the PSO algorithm has been successfully applied to problems in different areas, including antenna design, computer graphics visualization, biomedical applications, design of electrical networks, and many others [9]. Amongst the qualities that led to this popularity are its conceptual simplicity, ease of implementation and low computational costs. The algorithm works well with small swarms and tends to converge fast to a solution. There are several variants of PSO, including algorithms for discrete and continuous domains. The variant used in our work was the discrete PSO introduced by Kennedy [10]. The discrete variant of the PSO considers each bit as a dimension, with 2 possible values 0 or 1. Particle motion is just the toggling between these two values. There is a velocity value (V_{id}) associated with each dimension/bit; this value is randomly generated from a range of $[-4.0, 4.0]$ when the particle is created and iteratively updated (see equation 1) according to his previous best position (P_{id}), and the best position in the neighborhood (P_{gd}). φ_1 and φ_2 are random weights whose role is to provide diversity between individual learning and social influence.

$$\mathbf{V_{id}(t+1) = V_{id}(t-1) + \varphi_1(P_{id} - x_{id}(t-1)) + \varphi_2(P_{gd} - xi, d(t-1))} \quad (1)$$

To determine if a bit will toggle (see equation 2), a random number, ρ, is drawn from a uniform distribution ranging from 0 to 1, and is compared to a normalized value of the velocity associated with this dimension.

$$x_{i,d} = \begin{cases} 1 & \text{if } \rho < S(v_{id}(t)) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The sigmoid function (see equation 3) is used here to insure that the velocity's value is kept in the range of $[0.0, 1.0]$.

$$S(v_{id}) = \frac{1}{1 + exp(-v_{id})} \quad (3)$$

3 The Disambiguation Rules

One of the problems of the stochastic taggers, is that they tend to be dependent of the domain in which they are trained. Also the information used is in the form of probabilistic values, which are less comprehensible than data presented in the form of rules, and only contemplates context information. The other evolutionary taggers, as far as we know, also used stochastic information to guide the evolutionary process. However they were able to consider different context shapes in the decision process, due to the use of the evolutionary algorithm in opposition to the hidden Markov model used in the traditional stochastic taggers.

We believe that the optimization process necessary to perform the tagging could be improved if the information used to guide it is in the form of rules that capture some relevant aspects of the language. This aspects should include not only the word's context, but also information about some of its morphology features.

In this work we used a set of disambiguation rules found earlier by a discrete PSO discovery algorithm to help the optimization process. These rules guided the swarm particles towards the best tagging for a sentence. Since the tagging of unambiguous words is straightforward, we only used rules for the ambiguous tags of the corpora used in the experimental work. A set of disambiguation rules is given to the PSO-Tagger for each ambiguous tag. The rules follow the classification rules format, and, therefore, are no more than conditional clauses, involving two parts: the antecedent and the consequent. The former is the conjunction of logical tests, and the latter gives the class that applies to instances covered by the rule: `IF` $attrib_a$ `=` val_1 `AND` $attrib_b$ `=` val_2 `.....` `AND` $attrib_n$ `=` val_i `THEN` $class_x$.

Since each tag has its set of disambiguation rules, the rule's consequent was ignored. The antecedent takes into consideration six attributes concerning the word's context: the lexical categories of the third, second and first words to the left, and the lexical categories of the first, second, and third words to the right. In addition, nine attributes that capture some aspects of the words' morphology, are also considered: 'The word is capitalized?', 'The word is the first word of the sentence?', 'The word ends with ***ed***?', 'The word ends with ***ing***?', 'The word ends with ***es***?' , 'The word ends with ***ould***?' , 'The word ends with ***'s***?' , 'The word ends with ***s***?', and 'The word has numbers or '.' and numbers?'.

The possible values for each of the first six attributes are the values of the corpus tag set and the other nine are boolean attributes. Each rule has associated a measure of its quality that was computed during the discovery process. This value was given by the well known f_β-measure (see equation 4), that takes into consideration the precision (see equation 5) and the recall (see equation 6) of the rule.

$$F_\beta(X) = (1 + \beta^2) \times \frac{precision(X) \times recall(X)}{\beta^2 \times precison(X) + recall(X)} \tag{4}$$

$$precision(X) = \frac{TP}{TP + FP} \tag{5}$$

$$recall(X) = \frac{TP}{TP + FN} \tag{6}$$

where:

- TP - True Positives = number of instances covered by the rule that are correctly classified, i.e., its class matches the training target class;
- FP - False Positives = number of instances covered by the rule that are wrongly classified, i.e., its class differs from the training target class;
- FN - False Negatives = number of instances not covered by the rule, whose class matches the training target class.

The rules were extracted from the first 50000 examples of the Brown corpus.

4 PSO-Tagger

The PSO-Tagger was designed to receive as inputs a sentence, a set of disambiguation rules and a dictionary. The returned output is the input sentence with each of its words marked with the correct POS tag. The discrete PSO algorithm evolves a swarm of particles, that encode, each of them, a sequence of tags to mark the ambiguous words of the input sentence. Since we adopted the discrete version of the PSO algorithm, we used a binary representation for the particles. To encode each of the tags belonging to the tag set, used in the experimental work, we used a string of 5 bits. Therefore, a particle that proposes a tagging for a sentence with n ambiguous words will be represented by $n \times 5$ bits.

4.1 Representation

Each five bits of a particle encode a integer number that indexes a table with as much entries as the possible tags for the correspondent ambiguous word. If the integer number given by the binary string exceeds the table size, we use as index the remainder of the division by the table size value. As we said before, we intend to use the disambiguation rules, described in the previous section, to guide the evolution of the swarm particles. Since these rules have six attributes related with the word context, and other nine attributes concerning morphological properties of the words, we need to build, from the input sentence and from the tags proposed by the particles, the values of each one of the attributes contemplated in the rules' antecedents. Although the particles only propose tags for the ambiguous words, the tags of the unambiguous ones will be needed to extract the attributes' values. Thus, before optimization begins, the discrete PSO marks each of these words with the correspondent tag, by simple input dictionary lookup. So a particle completes the previous lookup based tagging, and provides a full marked sentence. This sentence can then be used to extract a set of instances composed by the 15 attributes, so that the disambiguation rules can be applied. This way, each pair w_i/t_i in the full annotated sentence results in a 16-tuple made by the 15 properties and by t_i. When there is no word in one of the positions contemplated in the context, we adopted the use of an extra tag named 'None'.

4.2 Particles' Evaluation

The quality of a particle is given by the tagging quality of the full input sentence. To evaluate the tagging of the sentence, we use the disambiguated rules to measure the quality of each instance extracted from the sentence. The quality of the overall tagging is given by the sum of the evaluation results for each instance. Let's consider t_i to be the class presented in the last position of the 16-tuple of instance $instance_k$. If R_{t_i} represents the set of disambiguation rules for the lexical category t_i, and $r_k \in R_{t_i}$ a disambiguation rule that covers the instance $instance_k$, then the quality value of $instance_k$ is given by the quality measure associated with rule r_k (see equation 7).

$$F(instance_k) = \begin{cases} Quality(r_k) & \text{if } r_k \text{ is found} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

If S_p is the set of all instances extracted from the annotated sentence determined by the particle p, the quality of particle p is given by equation 8.

$$Fitness(p) = \sum_{i \in S_p} F(i) \tag{8}$$

Although a particle only suggests tags for the ambiguous words in the sentence, the quality of the instances defined by the unambiguous words is affected by the tags established by the particle. Therefore, the overall tagging evaluation should also consider this instances.

5 Experimental Results

We developed our system in Python and used the resources available on the NLTK (Natural Language Toolkit) software package in our experiences. The NLTK package provides, among others, the Brown corpus and a sample of 10% of the Wall Street Journal (WSJ) corpus of the Penn Treebank. These corpora are the most frequently used to test taggers' performances and also the ones used in the approaches we mentioned earlier. The NLTK package also provides several Python modules to process those corpora.

As we said before, tagged corpora use many different conventions for tagging words. This means that the tag sets vary from corpus to corpus. To avoid this, we decided to use the *simplify_ tags=True* option of the *tagged_ sentence* module of NLTK corpus readers. When this option is set to *True*, NLTK converts the respective tag set of the corpus used to a uniform simplified tag set, composed by 20 tags. This simplified tag set was the one used by the PSO discovery algorithm to extract the disambiguation rules we used, and, for that reason, the one adopted here.

We tested our PSO-Tagger on 8300 words of the WSJ corpus of the Penn Treebank and on 22562 words of the Brown corpus. We ran the algorithm 20 times with a swarm of 10 and 20 particles during 50 and 100 generations. The results achieved are shown in table 1. As we can see, the best average accuracy on

Table 1. Results achieved on the Brown and WSJ corpora by the PSO-Tagger after 20 runs with a swarm of size 10 and 20, during 50 and 100 generations

Corpus	Swarm	Gen	Average	Stand. Dev.	Best
Brown	10	50	96.68	0.022	96.72
		100	96.67	0.028	96.72
	20	50	**96.7**	0.024	**96.75**
		100	96.69	0.024	96.73
WSJ	10	50	96.9	0.054	96.99
		100	**96.91**	0.054	**97.04**
	20	50	96.88	0.053	96.99
		100	96.91	0.031	96.98

Table 2. Results achieved by the PSO-Tagger on the WSJ corpus and on the Brown corpus, along with the results achieved by the approaches more similar to the one presented here

Corpus	Tagger	Training Set	Test Set	Average	Best
Brown	PSO-Tagger	50000	22562	96.7	**96.75**
	GA-Tagger [2]	185000	2500	-	95.4
	GA-Tagger [4]	165276	17303	96.37	96.67
	PGA-Tagger [4]	165276	17303	96.61	96.75
WSJ	PSO-Tagger	none	8300	96.91	**97.04**
	Wilson's Tagger [6]	600000	none	-	89.8
	Brill's Tagger [5]	600000	150000	-	97.2
	PGA-Tagger [4]	554923	2544	-	96.63

the WSJ corpus was 96.91% and the best average accuracy on the Brown corpus was 96.72%. The best results on the Brown corpus were achieved with a swarm of 20 particles over 50 generations. A maximum accuracy of 96.75% was found. A swarm of 20 particles over 100 generations gave the best results for the WSJ corpus. In this case, the best tagging found had an accuracy of 97.04%. Both results allow us to conclude that the PSO-Tagger usually finds a solution very quickly. Like we said before, this algorithm works well with small swarms and tends to converge fast to a solution. Table 2 presents the best results achieved by the approaches we mentioned earlier, along with the best ones achieved by the PSO-Tagger. Observing table 2, we can see that the PSO-Tagger accuracy is very promising, since it is among the best values presented.

Naturally, the difficulty level of the tagging task depends on the number of ambiguous words of the sentence we want to tag. Although it is possible to construct sentences in which every word is ambiguous [11], such as the following: "Her hand had come to rest on that very book."; those situations are not the most common. After counting the number of ambiguous words that appear in the sentences of the 10% of the Brown corpus we reserved for testing the tagger, we observed that, in

average, there are 6.9 ambiguous words per sentence. This explain the considerable low number of particles and generations needed to achieve a solution. We could argue that in those conditions the use of a PSO algorithm is unnecessary, and that a exhaustive search could be applied to solve the problem. However, we can not ignore the worst case scenario, where, like we see above, all the words, or a large majority of the words, on a very long sentence may be ambiguous. Furthermore, we observed that the sentence average size of the Brown corpus is of 20.25 tokens, with a maximum of 180. The largest number of ambiguous words on a sentence belonging to this corpus is 68. Even for the smallest degree of ambiguity, with only two possible tags for each word, we have a search space of 2^{68}, which fully justifies the use of a global search algorithm such as a PSO.

The results achieved show that there are no significant differences on the accuracy obtained by the tagger on the two test sets. At this point, it is important to emphasize that the disambiguation rules used on the tagger were extracted from a subset (different from the test set used in this experiments) of the Brown corpus. Which bring us to the conclusion that the learned rules are generic enough to be used on different corpora, and are not domain dependent.

6 Conclusions

We described a new evolutionary approach to the POS tagging problem that achieved competitive results when compared to the ones obtained by previously used methods (see table 2). Although there are other approaches to this task based on evolutionary algorithms, in particular genetic algorithms, as far as we know this is the first attempt that uses a PSO algorithm to tackle the POS tagging problem. Our method also differs from previous ones on the information model used to guide the evolutionary process. More specifically, in this work, we used a set of disambiguation rules, including morphological information, in opposition to the stochastic data that is usually adopted in the evolutionary approaches we have found in the literature. We believe that this approach brings an important level of generalization to the model, which results in good tagging performances even when applied to other corpora.

The PSO-Tagger proved to be capable of tackling the combinatorial optimization problem, performing the tagging task with good results, while using limited resources in terms of swarm size and number of generations. Although we consider our results to be promising, we are aware of the necessity of evaluating our approach with a larger tag set and of applying it to more corpora. We intend to test the tagger on other languages, as well. Finally, we also think that the overall evolutionary approach we developed for the POS tagging problem could be successfully applied to other disambiguation problems, like the named-entity recognition problem.

References

1. Brants, T.: Tnt: a statistical part-of-speech tagger. In: Proceedings of the Sixth Conference on Applied Natural Language Processing, ANLC 2000, pp. 224–231. Association for Computational Linguistics, Stroudsburg (2000)
2. Araujo, L.: Part-of-Speech Tagging with Evolutionary Algorithms. In: Gelbukh, A. (ed.) CICLing 2002. LNCS, vol. 2276, pp. 230–239. Springer, Heidelberg (2002)
3. Araujo, L., Luque, G., Alba, E.: Metaheuristics for Natural Language Tagging. In: Deb, K., Tari, Z. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 889–900. Springer, Heidelberg (2004)
4. Alba, E., Luque, G., Araujo, L.: Natural language tagging with genetic algorithms. Inf. Process. Lett. 100(5), 173–182 (2006)
5. Brill, E.: Transformation-based error-driven learning and natural language processing: a case study in part-of-speech tagging. Comput. Linguist. 21, 543–565 (1995)
6. Wilson, G., Heywood, M.: Use of a genetic algorithm in brill's transformation-based part-of-speech tagger. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, GECCO 2005, pp. 2067–2073. ACM, New York (2005)
7. Nogueira dos Santos, C., Milidiú, R.L., Rentería, R.P.: Portuguese Part-of-Speech Tagging Using Entropy Guided Transformation Learning. In: Teixeira, A., de Lima, V.L.S., de Oliveira, L.C., Quaresma, P. (eds.) PROPOR 2008. LNCS (LNAI), vol. 5190, pp. 143–152. Springer, Heidelberg (2008)
8. Steven Bird, E.K., Loper, E.: Natural Language Processing with Python. O'Reilly Media (2009)
9. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. App., 4:1–4:10 (January 2008)
10. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
11. Hindle, D.: Acquiring disambiguation rules from text (1989)

Training Support Vector Machines with an Heterogeneous Particle Swarm Optimizer

Arlindo Silva[1] and Teresa Gonçalves[2]

[1] Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco
arlindo@ipcb.pt
[2] Universidade de Évora
tcg@uevora.pt

Abstract. Support vector machines are classification algorithms that have been successfully applied to problems in many different areas. Recently, evolutionary algorithms have been used to train support vector machines, which proved particularly useful in some multi-objective formulations and when indefinite kernels are used. In this paper, we propose a new heterogeneous particle swarm optimization algorithm, called scouting predator-prey optimizer, specially adapted for the training of support vector machines. We compare our algorithm with two other evolutionary approaches, using both positive definite and indefinite kernels, on a large set of benchmark problems. The experimental results confirm that the evolutionary algorithms can be competitive with the classic methods and even superior when using indefinite kernels. The scouting predator-prey optimizer can train support vector machines with similar or better classification accuracy than the other evolutionary algorithms, while requiring significantly less computational resources.

Keywords: particle swarm optimization, heterogeneous particle swarms, support vector machines, non PSD kernels.

1 Introduction

Kernel methods are data analysis techniques, whose strategy is to map the original data into a feature space, where existing patterns can be discovered using simple linear relations [1]. This process is carried out on a modular basis, as each step is performed by separate components. The data mapping component is defined implicitly through a kernel function. The choice of the kernel function depends on the problem's characteristics and knowledge about the patterns that are expected to be found. Linear patterns are then searched in the resulting feature space, using problem independent learning algorithms.

The best known representatives of these methods are support vector machines (SVMs), which classify new data by comparing it with a learned hyper-plane that maximizes a margin between data points of different classes [2]. The remarkable success with which these methods have been applied to many areas is the result of their specific properties: low computational cost; robustness, with

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 100–109, 2013.

solid theoretical bases in statistical learning; and generality, since the choice of an appropriate kernel function allows the algorithm to learn non-linear decision functions and deal with non-vectorial and even heterogeneous data.

Despite the success of this approach, the application of a SVM to a new problem still presents a number of difficulties. A kernel function must be chosen and its parameters optimized. A real parameter C must also be chosen to balance error and capacity in the SVM. If a new kernel function has to be developed, care must be taken to ensure the kernel is positive semi definite (PSD), since training relies on quadratic programming based techniques, where a unimodal concave function is optimized. These issues have been addressed using both analytic techniques and heuristic search algorithms. Recently, several approaches based on evolutionary methods, such as genetic algorithms (GA), genetic programming (GP) and particle swarm optimization algorithms (PSO) have been proposed. These algorithms are advantageous when search and/or optimization is done in complex, non-vectorial spaces, or when the function to optimize is multi-modal, with many local optima in alternative to a single solution. In this paper, we deal with evolutionary approaches to the training of support vector machines.

There are three main reasons that make the evolutionary training of SVMs an interesting problem. The most important concerns the possibility of using evolutionary algorithms to train SVMs using indefinite kernels [3]. Learning with indefinite kernels is an important research area, because traditional methods are not guaranteed to find the global optimum on the resulting optimization problem; proving a new kernel to be PSD can be a difficult task; some kernels that are proven non PSD, e.g. the sigmoid kernel, can be of practical interest [4]; there were promising empirical results reported for SVMs using indefinite kernels [4]; some kernel learning approaches (including GP based methods) return kernels that are not guaranteed to be PSD [5]. The second reason is related with the recent proposal of multi-objective evolutionary SVM formulations [6], which allow the independent optimization of error and model complexity, i.e., the trade-off parameter C is not needed. Finally, this is an interesting practical problem for evolutionary computation, with an objective function that is high-dimensional, multi-modal (for indefinite kernels) and non-separable.

In this paper, we propose the use of an heterogeneous particle swarm optimizer, called scouting predator-prey optimizer (SPPO), to train the support vector machines. We empirically compare this algorithm with the best evolutionary based approach found in the field literature [7], the canonical constricted version of the PSO and two standard training methods. The comparison is done over a large set of benchmark datasets, including 7 real world and 3 synthetic datasets. The algorithms are tested using two different kernels, the radial basis function and a non positive definite kernel, the Epanechnikov kernel.

2 Previous Work

The optimization of the kernel and C parameters are the issues that have attracted the most attention from the evolutionary computation community (see,

e.g., [8]). Some approaches, of which [9] is a recent example, use genetic programming to evolve the kernel best suited for a given problem. While these methods are very computationally expensive, they also frequently find new kernels with better performance than the more usual ones. A difficulty common to these approaches is how to deal with the evolved kernels, when they are non PSD. Initial research in the evolutionary training of support vector machines has been centered on the training of SVMs with positive definite kernels. The first approach reported in the literature was a hybrid approach that combined a linear particle swarm optimizer with traditional decomposition based methods [10]. It had some severe problems and experimental results were very limited. An approach based on a genetic algorithm has been used to optimize the primal problem instead of the more common dual version [11], which seriously limits its applicability.

The most significant work in SVM evolutionary training was based on the use of evolution strategies (ES). Mierswa compared several ES based algorithms with a standard PSO optimizer on 6 benchmark problems and found that the evolutionary algorithms' performance was competitive with the traditional methods [7], but that the PSO did not achieve as lower classification errors as the other algorithms. The best ES approach was then used as the optimization method for a new multi-objective SVM formulation [6], allowing the simultaneous independent optimization of the classification error and model complexity. The same method was also applied to the training of a SVM with an Epanechnikov kernel [3], in what was the first described application of an evolutionary training algorithm to the optimization of SVMs with a non PSD kernel. The evolutionary method was reported to have achieved better optimization and classification results in several benchmark problems, when compared with a traditional quadratic programming based method.

3 Support Vector Machines

In their most common formulation [1,2,12], support vector machines are classification mechanisms, which, given a training set $T = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \{\pm 1\}$, assuming n exemples with m real attributes, learn a hyperplane $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$, with $\mathbf{w} \in \mathbb{R}^m$ and $b \in \mathbb{R}$, which completely separates the example labels as -1 from the ones labeled as $+1$. Using this hyperplane, a new instance $\mathbf{x}$ is classified using $f(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$.

The maximization of the distance, or margin, between the discriminating hyperplane and the closest examples, is a characteristic of large margin methods, of which support vector machines are an instance. This maximization reduces the so-called structural risk, which is related to the quality of the decision function. Support vector machines therefore try to minimize the structural risk, which comprises not only the empirical risk, but also a measure of the classifier's quality. In general, these methods seek to avoid the over-adjustment of the classifier, by giving preference to classification functions which appear to be the most promising in terms of their ability to generalize for new data. The most discriminating hyperplane can be computed by solving the following maximization problem:

$$\text{maximize } \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j k(\mathbf{x}_i, \mathbf{x}_j), \tag{1}$$

$$\text{subject to } \sum_i \alpha_i y_i = 0 \tag{2}$$

$$\text{and } \forall i : 0 \leqslant \alpha_i \leqslant C \tag{3}$$

Support vector machines are extended to the non-linear case by implicitly mapping the data to a secondary space with higher dimensionality - the feature space - where a linear separation is possible. This mapping is achieved by using a kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ in equation (1) instead of the product $\langle \mathbf{x}_i, \mathbf{x}_j \rangle$. The most common kernel function is the radial basis function, but many others are used. If the kernel is positive definite, the optimization problem is a quadratic problem with a concave optimization function and several algorithms are available to solve it efficiently, with the most popular implementations being mySVM and LIBSVM. If, however, the kernel is not positive definite, the problem is not guaranteed to possess a single optimum and these algorithms can become trapped in local optima, failing to find the best solution.

Central to the wide acceptance of SVMs are their sound theoretical foundations and clear geometric interpretation. SVMs work on Euclidean feature spaces implicitly defined by the kernel functions. They classify patterns using an optimal hyperplane computed by maximizing the margin to the nearest training examples - the support vectors. Unfortunately, this interpretation fails to hold when non positive definite kernels are used, even if experimental results show that these kernels can result in successful classifiers.

Haasdonk, however, has proposed an alternative interpretation for SVMs based on non positive definite kernels, which accounts for their good experimental results [13]. In this interpretation, SVMs are optimal hyperplane classifiers, not by margin maximization, but by minimization of distances between convex hulls in pseudo-Euclidean spaces. Both this work and the one by [4] conclude that traditional methods, e.g. LIBSVM, can converge to a stationary point of the non concave optimization problem that results from the use of indefinite kernels. Obviously, there is no guarantee that this point is the global optimum, which leads us to the interest in using heuristic global optimization methods, like the evolutionary algorithms we discuss here. An in-depth discussion of the relevance of learning with non PSD kernels can be found in [5], while the training of SVM using these kernels is thoroughly discussed in [3,4,13].

4 The Scouting Predator-Prey Optimiser

Particle swarm optimizers [14,15] can be an obvious answer to the problem of optimizing SVMs with non positive definite kennels, since they are population based global optimization algorithms with successful application to hard optimization problems with many optima [16]. We recently proposed a new heterogenous particle swarm algorithm, called scouting predator-prey optimizer (SPPO),

which showed good performance, even on hard optimization problems [17]. We also proposed the use of scout particles to improve a swarm optimizer by using problem specific knowledge. Here, we will describe a version of the SPPO specifically tailored to the training of SVMs.

In particle swarm optimization, each swarm member is represented by three m-size vectors, assuming an optimization problem $f(\mathbf{x})$ in $\mathbb{R}^m$. For each particle i we have a $\mathbf{x}_i$ vector that represents the current position in the search space, a $\mathbf{p}_i$ vector storing the best position found so far and a third vector $\mathbf{v}_i$ corresponding to the particle's velocity. For each iteration t of the algorithm, the current position $\mathbf{x}_i$ of every particle i is evaluated by computing $f(\mathbf{x}_i)$. Assuming a maximization problem, $\mathbf{x}_i$ is saved in $\mathbf{p}_i$ if $f(\mathbf{x}_i) > f(\mathbf{p}_i)$, i.e. if $\mathbf{x}_i$ is the best solution found by the particle so far. The velocity vector $\mathbf{v}_i$ is then computed using equation (4) and used to update the particle position using equation (5).

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + \mathbf{u}(0,\phi_1) \otimes (\mathbf{p}_i^t - \mathbf{x}_i^t) + \mathbf{u}(0,\phi_2) \otimes (\mathbf{p}_g^t - \mathbf{x}_i^t) \tag{4}$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^t \tag{5}$$

In equation (4) $(\mathbf{p}_i^t - \mathbf{x}_i^t)$ represents the distance between a particle and its best position in previous iterations and $(\mathbf{p}_g^t - \mathbf{x}_i^t)$ represents the distance between a particle and the best position found by the particles in its neighborhood (which can be the complete swarm), stored in $\mathbf{p}_g^t$. $\mathbf{u}(0,\phi_1)$ and $\mathbf{u}(0,\phi_2)$ are random number vectors with uniform distributions between 0 e ϕ_1 and 0 e ϕ_2, respectively. w is a weight associated with the velocity in the previous iteration, which usually decreases linearly during the algorithm execution. $\otimes$ is a vector component-wise multiplication.

4.1 The Predator Effect

One of the limitations of the standard particle swarm algorithm is its inability to introduce diversity in the swarm after it has converged to a local optimum. Since there is no mechanism similar to a mutation operator, and changes in $\mathbf{x}_i$ are dependent on differences between the particles' positions, as the swarm clusters around a promising area in the search space, so does velocity decreases and particles converge to the optimum. This is the desirable behavior if the optimum is global, but, if it is a local optimum, there is no way to increase velocities again and allow the swarm to escape to a new optimum. We use a predator-prey effect in SPPO to alleviate this problem. The predator particle's velocity $\mathbf{v}_p$ is updated using equation (6), oscillating between the best particle's best position and the best particle's current position. This update rule makes the predator effectively chase the best particle in the search space.

$$\mathbf{v}_p^{t+1} = w\mathbf{v}_p^t + \mathbf{u}(0,\phi_1) \otimes (\mathbf{x}_g^t - \mathbf{x}_p^t) + \mathbf{u}(0,\phi_2) \otimes (\mathbf{p}_g^t - \mathbf{x}_p^t) \tag{6}$$

The role of the predator particle in the SPPO algorithm is to introduce a perturbation factor in the swarm and to guarantee that this disturbance increases

as the swarm converges to a single point. To achieve this, we add a perturbation to a particles's velocity in dimension j, as defined by equation 7, where $u(-1,1)$ and $u(0,1)$ are random numbers uniformly distributed between the arguments, x_{max} and x_{min} are, respectively the upper and lower limit to the search space and r is the user defined perturbation probability.

$$v_{ij}^t = v_{ij}^t + u(-1,1)|x_{max} - x_{min}|, \text{if } u(0,1) < r \exp^{-|x_{ij} - x_{pj}|} \tag{7}$$

From equation 7 follows that a random perturbation is added to the velocity value in dimension j with a probability that depends on the particles's distance to the predator in that dimension. This probability is maximum (r) when that distance is 0, but rapidly decreases if the particle escapes the predator. Since the predator chases the best particle, the perturbation in the swarm is more likely when all the particles are very near, i.e. during the exploitation phase, and becomes almost inexistent when the particles are far apart. This mechanism allows for particles to escape and find new optima far from the current attractor even in the last phases of exploitation.

4.2 Scout Particles

Scout particles, or scouts, are a subset of the swarm that implement exploration strategies different from the one used by the main swarm. They can be used to introduce improvements to the global algorithm, e.g. a local search sub-algorithm, or to implement problem dependent mechanisms to better adapt the algorithm to a specific problem. In this work, we will use two scout particles to tailor the SPPO to the specific problem of training SVMs. The first scout is a local search particle which, from previous work [17], we know can be used to increase the convergence speed without compromising the final results. For this scout we choose the best particle at each iteration and perform a random mutation on one of its dimensions j using equation (8), where $n(0,\sigma^2)$ is a a random number drawn from a normal distribution with average 0 and standard deviation σ. $\mathbf{p}_g$ is updated to the new $\mathbf{p}'_g$ only if $f(\mathbf{p}'_g) > f(\mathbf{p}_g)$. σ is set to $x_{max}/10$. This mechanism allows for a local search to be made around $\mathbf{p}_g$ over time.

$$p'_{gj} = p_{gj} + n(0,\sigma) \tag{8}$$

The second scout particle uses specific problem knowledge to accelerate the training process. Since we know that in the final solution only the few α_i corresponding to support vectors will be different from 0, and that, from these, most will be C, we will, at every iteration move the scout particle, in a random dimension, to an extreme of the search space, with an 80% probability of that extreme being 0 and 20% of being C. This scout will consequently explore the border of the search space, where we know the solution should be in a large majority of dimensions. Scout particles are updated prior to the main update cycle of the swarm, where they can cumulatively be updated using equations (4) and (7).

5 Experimental Results

In the first set of experiments, we tested three evolutionary approaches, including the best previous ES based approach [7], a constricted PSO and the SPPO algorithm previously described, against the two most popular quadratic programming based methods, MySVM and LIBSVM, in a set of 10 benchmark problems, 3 of which are synthetically generated and the remaining 7 are real world benchmark problems. Table 1 lists the datasets' names, source, number of attributes n and number of instances m. e_d is the error for a classifier that always returns the most frequent class. The kernel used in all experiences was the radial basis function and its parameter σ was previously optimized for the MySVM method using a grid search (See Table 1). We used $C = 1$ for all datasets.

Table 1. Dataset and kernel parameters

Dataset	Source	n	m	e_d	σ	σ_E	d
Checkerboard	Synthetic	1000	2	48.60	100	0.92	6.54
Spirals	Synthetic	500	2	50.00	100	0.12	3.84
Threenorm	Synthetic	500	2	50.00	1	61.60	9.38
Credit	UCI MLR	690	14	44.49	0.001	564.62	0.65
Diabetes	UCI MLR	768	8	34.90	0.1	220.51	4.87
Ionosphere	UCI MLR	351	34	35.90	0.1	2.44	7.48
Liver	UCI MLR	345	6	42.03	1	61.59	6.90
Lupus	StatLib	87	3	40.23	0.1	241.63	7.42
Musk	UCI MLR	476	166	43.49	0.1	63.12	6.93
Sonar	UCI MLR	208	60	46.63	0.1	61.63	6.90

The evolutionary approaches were implemented to search for the vector α that maximizes equation (1). Each α_i was limited to the interval $[0, C]$, respecting restrictions (2). Fixing $b = 0$ during optimization makes restriction 3 disappear, increasing the efficiency of the optimizers. After the optimization is done, a value for b can be computed using the restriction. This is a common technique in other approaches, since the fixing of just one variable in the optimization problem is a mild disadvantage, at least for high-dimensional problems [12]. Since the used kernel is positive definite, the problem is unimodal and the main difficulties for the evolutionary approaches are the high dimensionality of the optimization function and the fact that many of the α_i are usually 0 or C, placing the solution in the frontier of the search space in many dimensions. The evolutionary algorithms were run for 500 iterations with 20 individuals/particles, except the SPPO, which only used 18 particles to compensate for the extra evaluations of the scout particles. Evaluation is done using 20-fold cross-validation. Experiences were run using RapidMiner software [18] with additional operators.

Table 2 presents average error rates and respective standard deviations for all pairs algorithm/dataset. We found that the best evolutionary approaches performed as well as the classical methods, both in terms of accuracy (error percentage) and robustness (standard deviation). Only the simple PSO performed poorer, which is compatible with the findings in [7]. Since the classical approaches are significantly faster than the evolutionary ones, there's no particular reason to prefer the later for the training of SVMs with PSD kernels. These results are still useful to demonstrate some debilities of the standard PSO and to demonstrate that the SPPO is the first competitive swarm intelligence based approach to the training of support vector machines.

Table 2. Experimental results (error percentage) using the RBF kernel

Dataset	MySVM	LIBSVM	ES	PSO	SPPO
Checkerboard	5.6 (2.6)	5.6 (3.2)	5.5 (3.0)	8.1 (4.3)	5.7 (3.4)
Spirals	0.4 (1.2)	0.2 (0.9)	0.6 (1.4)	2.2 (2.0)	0.6 (1.9)
Threenorm	15.0 (6.3)	14.8 (4.9)	14.6 (6.5)	14.2 (5.0)	14.8 (8.1)
Credit	14.5 (6.5)	14.5 (4.5)	14.5 (6.5)	13.8 (6.4)	14.4 (5.7)
Diabetes	22.4 (5.0)	22.5 (4.8)	23.7 (5.5)	29.8 (5.6)	23.3 (7.4)
Ionosphere	6.8 (5.9)	6.2 (5.6)	7.1 (5.7)	24.2 (9.1)	7.1 (5.7)
Liver	29.9 (11.9)	28.4 (10.5)	29.3 (11.9)	31.7 (10.4)	28.7 (11.1)
Lupus	26.0 (22.4)	24.8 (22.8)	25.0 (22.4)	26.0 (24.5)	25.0 (16.6)
Musk	7.8 (5.2)	7.5 (4.5)	9.9 (6.0)	11.1 (6.9)	8.6 (7.4)
Sonar	14.4 (7.6)	16.0 (11.8)	14.4 (9.9)	15.3 (10.2)	14.3 (13.1)

In the second set of experiments, we investigate how the best evolutionary algorithms compare with one of the standard approaches when training SVMs with a non PSD kernel, i.e., in a multimodal optimization problem. We use the same datasets, but the RBF is substituted by an Epanechnikov kernel, which can be proved to be indefinite. C was set to 1 and kernel parameters are presented in Table 1. Since we observed in the convergence graphs of the previous experiences that the evolutionary algorithms converged a lot sooner than the allotted 500 generations, we reduced the iteration limit to 100 (150 for the synthetic problems). In Table 3 we present the classification error, and, for the evolutionary approaches, the average best value found for the objective function, $f(\alpha^*)$.

This second set of results allows us to draw several conclusions. First, all algorithms were able to learn with the non-PSD kernel. In fact, for two of the datasets, Lupus and Sonar, the best overall results were obtained using the Epanechnikov kernel, in both cases using the SPPO algorithm. Second, with the lower iteration limit, there is a large difference, both in classification accuracy and best $f(\alpha^*)$, between the evolutionary approaches. This leads us to conclude that the SPPO needs significantly less function evaluations to achieve similar (or superior) classification accuracy, when compared with the best ES

based approach. And, finally, we can see four datasets for which the SPPO performs better than the MySVM algorithm (results are similar for the rest). Since, from the previous experiments, we know that the algorithms are able to obtain identical results when using the same parameters in the concave optimization problems, these data apparently illustrate situations where, for a multimodal problem, the quadratic based approach fails to converge to the global optimum. This result confirms our proposition that evolutionary approaches can be useful tools in the training of SVMs when using non PSD kernels.

Table 3. Experimental results (error percentage) using the Epanechnikov kernel

Dataset	MySVM	ES ($f(\alpha^*)$)	ES	SPPO ($f(\alpha^*)$)	SPPO
Checkerboard	6.5 (4.6)	-304.1 (37.6)	8.2 (4.0)	60.9 (12.3)	7.5 (4.9)
Spirals	11.0 (6.5)	163.2 (3.2)	19.2 (7.8)	188.5 (2.5)	**7.8 (5.6)**
Threenorm	14.0 (7.5)	31.0 (11.9)	15.0 (6.9)	132.5 (4.7)	14.4 (6.6)
Credit	14.4 (5.5)	254.8 (6.2)	14.2 (6.5)	299.6 (5.5)	13.6 (4.4)
Diabetes	24.8 (6.2)	-62.4 (197.2)	29.4 (6.6)	297.5 (7.8)	25.2 (8.1)
Ionosphere	26.9 (10.6)	79.8 (3.9)	24.0 (9.3)	99.8 (1.6)	**16.1 (9.0)**
Liver	40.8 (3.7)	175.1 (5.5)	35.9 (12.1)	224.1 (4.7)	**35.4 (10.6)**
Lupus	27.8 (15.8)	48.6 (1.9)	24.0 (15.9)	58.4 (1.3)	**21.5 (18.8)**
Musk	9.6 (5.7)	104.5 (2.4)	11.8 (7.1)	118.4 (2.1)	9.5 (5.5)
Sonar	12.4 (11.4)	175.1 (5.5)	12.8 (10.8)	224.1 (4.7)	11.9 (9.6)

6 Conclusions

In this paper we proposed the first known particle swarm based approach to the problem of training SVMs with non PSD kernels. Our algorithm is a specially tailored version of the scouting predator prey algorithm [17], an heterogeneous particle swarm optimizer. To improve the algorithm performance in this particular problem, we embedded two scout particles in the algorithm, one to perform a local search and another to take advantage of specific problem knowledge. We compared our algorithm with the best known evolutionary approach to this task, as well as with two popular classical SVM training algorithms, using both a PSD and a non PSD kernel. The experimental results supported the assertions made in [3], since the evolutionary approaches, most specifically the SPPO, were able to outperform the classical method on several benchmark problems, when training the SVMs with the non PSD kernel. Regarding the evolutionary approaches, the results show that the SPPO can achieve significantly better values for the optimization function, with corresponding similar or better classification accuracy, than the ES based approach, for the same number of function evaluations.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge Univ. Press, Cambridge (2004)
2. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
3. Mierswa, I., Morik, K.: About the non-convex optimization problem induced by non-positive semidefinite kernel learning. Advances in Data Analysis and Classification 2, 241–258 (2008)
4. Lin, H.T., Lin, C.J.: A study on sigmoid kernel for svm and the training of non-psd kernels by smo-type methods. Technical report, National Taiwan University, Taipei, Department of Computer Science and Information Engineering (2003)
5. Ong, C.S., Mary, X., Canu, S., Smola, A.J.: Learning with non-positive kernels. In: Proceedings of the Twenty-First International Conference on Machine Learning, ICML 2004, ACM, New York (2004)
6. Mierswa, I.: Controlling overfitting with multi-objective support vector machines. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, GECCO 2007, pp. 1830–1837. ACM, New York (2007)
7. Mierswa, I.: Evolutionary learning with kernels: a generic solution for large margin problems. In: GECCO 2006: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, pp. 1553–1560. ACM, New York (2006)
8. Samanta, B., Nataraj, C.: Application of particle swarm optimization and proximal support vector machines for fault detection. Swarm Intelligence 3, 303–325 (2009)
9. Gilsberts, A., Metta, G., Rothkrantz, L.: Evolutionary optimization of least-squares support vector machines. In: Data Mining. Annals of Information Systems, vol. 8, pp. 277–297. Springer, US (2010)
10. Paquet, U., Engelbrecht, A.: Training support vector machines with particle swarms. In: Proceedings of the International Joint Conference on Neural Networks, vol. 2, pp. 1593–1598 (2003)
11. Stoean, R., Preuss, M., Stoean, C., Dumitrescu, D.: Concerning the potential of evolutionary support vector machines. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 1436–1443 (2007)
12. Burges, C.J.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2, 121–167 (1998)
13. Haasdonk, B.: Feature space interpretation of svms with indefinite kernels. IEEE Transactions on Pattern Analysis and Machine Intelligence 27, 482–492 (2005)
14. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
15. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1, 33–57 (2007)
16. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. App., 4:1–4:10 (January 2008)
17. Silva, A., Neves, A., Gonçalves, T.: An Heterogeneous Particle Swarm Optimizer with Predator and Scout Particles. In: Kamel, M., Karray, F., Hagras, H. (eds.) AIS 2012. LNCS, vol. 7326, pp. 200–208. Springer, Heidelberg (2012)
18. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: Yale: Rapid prototyping for complex data mining tasks. In: Proceedings of the 12th International Conference on Knowledge Discovery and Data Mining, KDD 2006 (2006)

Fitness Landscape-Based Characterisation of Nature-Inspired Algorithms

Matthew Crossley, Andy Nisbet, and Martyn Amos*

School of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, Manchester M15GD, UK
{m.crossley,a.nisbet,m.amos}@mmu.ac.uk

Abstract. A significant challenge in nature-inspired algorithmics is the identification of specific characteristics of problems that make them harder (or easier) to solve using specific methods. The hope is that, by identifying these characteristics, we may more easily predict which algorithms are best-suited to problems sharing certain features. Here, we approach this problem using fitness landscape analysis. Techniques already exist for measuring the "difficulty" of specific landscapes, but these are often designed solely with evolutionary algorithms in mind, and are generally specific to discrete optimisation. In this paper we develop an approach for comparing a wide range of continuous optimisation algorithms. Using a fitness landscape generation technique, we compare six different nature-inspired algorithms and identify which methods perform best on landscapes exhibiting specific features.

1 Introduction

Inspired by the foundational work of Wolpert and Macready [1], practitioners have long sought to better understand the relationship between problems and solution methods (i.e., algorithms). Here, we are particularly interested in the question of which algorithm is *best-suited* to a particular problem, and the process of addressing this has been described by some as a "black-art" [2].

Although theoretical studies in this area have yielded useful results, the *experimental analysis* of algorithms is receiving increasing attention. As Morgan and Gallagher point out [3], this approach is *scalable* in that it readily admits newly-described algorithms, and it is now an area of research that is supported by a number of high-profile competitions and libraries of benchmark test problems.

The fundamental properties of a problem's *search landscape* underpin much work in experimental analysis, and the use of landscape/test case generators [3–7] has been proposed as one way in which we might effectively assess algorithms against problem instances.

In this paper we examine six different nature-inspired algorithms by testing them against a number of different randomized landscapes with several different

* Matthew Crossley is supported by a Ph.D. studentship from the Dalton Research Institute, MMU. The authors thank David Corne for useful discussions.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 110–119, 2013.

properties (e.g., ruggedness). This gives a much richer picture of their relative strengths and weaknesses, compared to simply using the "difficulty" of a landscape [8].

The rest of the paper is organized as follows: in Section 2 we give a brief overview of previous work, before describing our testing methodology in Section 3. We then present our experimental results in Section 4, before concluding in Section 5 with a discussion of our findings.

2 Previous Work

The use of algorithms inspired by physical or natural processes is now well-established in the field of optimisation [9]. As the number of such algorithms grows year-on-year, there is a pressing need to better understand their properties, in order that practitioners may make informed decisions about which method is best-suited to a particular problem, under certain conditions. Although analytical methods have been successfully applied to nature-inspired methods [10] [11], their "real world" applicability is not clear, as they often rely on significant assumptions and/or simplifications.

In what follows, we take an *experimental* approach [12] to studying the selected algorithms, using an established landscape generation technique [4]. As Morgan and Gallagher observe, "In a general sense, an algorithm can be expected to perform well if the assumptions that it makes, either explicit or implicit, are well-matched to the properties of the search landscape or solution space of a given problem or set of problems" [3]. We therefore seek to investigate the performance of several algorithms on a number of types of *fitness landscape* with specific properties or characteristics. This approach is preferred by Hooker to the use of benchmark problems, because the latter "differ in so many respects that it is rarely evident why some are harder than others, and they may yet fail to vary over parameters that are key determinants of performance. It is better generate problems in a controlled fashion... The goal is not to generate realistic problems, which random generation cannot do, but to generate several problem sets, each of which is homogeneous with respect to characteristics that are likely to affect performance" [13].

The fitness landscape approach has been successfully applied to the study of various nature-inspired algorithms [14–16]. Indeed, to our knowledge, landscape analysis of nature-inspired algorithms has been largely *restricted* to evolutionary methods. In this paper we broaden this work *considerably*, by considering several classes of natural algorithms (social, evolutionary and physical). Overall, we study six different nature-inspired methods, as well as stochastic hill-climbing as a baseline algorithm. Our empirical approach is informed by previous work [17] [18], which emphasises the need to establish a rigorous framework for experimental algorithmics. In the next Section, we describe in detail our methodology.

3 Methodology

3.1 Algorithm Selection

We select, for comparison, a number of nature-inspired algorithms that are commonly applied to continuous function optimisation. These may be classified [19] as either *social*, *evolutionary* or *physical*. The social algorithms we select are Bacterial Foraging Optimisation Algorithm (BFOA) [20], Bees Algorithm (BA) [21], and Particle Swarm Optimisation (PSO) [22]. The evolutionary algorithms selected are Genetic Algorithms (GA) [23] and Evolution Strategies (ES) [24], and physical algorithms are represented by Harmony Search (HS) [25]. We also include random search (RS) and stochastic hill climbing (SHC) as "baseline" algorithms.

We note that the references supplied above for each algorithm may serve simply as an example of their *application*, rather than their precise *implementation*. In terms of implementation, we heed the observation that "Ideally, competing algorithms would be coded by the same expert programmer and run on the same test problems on the same computer configuration" [12]. With that in mind, we use only implementations provided by Brownlee to accompany [26]. The limited space available prevents a complete description of each algorithm, but full implementation details are in [26], which is freely available and contains the source code used here.

3.2 Optimisation Problem Characteristics

As Morgan and Gallagher explain [3], their Max-Set of Gaussians (MSG) method [4] is a "randomised landscape generator that specifies test problems as a weighted sum of Gaussian functions. By specifying the number of Gaussians and the mean and covariance parameters for each component, a variety of test landscape instances can be generated. The topological properties of the landscapes are intuitively related to (and vary smoothly with) the parameters of the generator." By manipulating these parameters, we obtain landscapes with different *characteristics*. This allows us to investigate the performance of our selected algorithms on landscapes with different features, and to identify which characteristics pose the greatest challenge. As Morgan and Gallagher observe, "Different problem types have their own characteristics, however it is usually the case that complementary insights into algorithm behaviour result from conducting larger experimental studies using a variety of different problem types" [3]. We now describe the different characteristics (corresponding to problem types) under study in this paper.

Ruggedness of a landscape is often linked to its difficulty [8], and factors affecting this include (1) the *number* of local optima [27], and (2) *ratio* of the fitness value of local optima to the global optimal value [28] [14]. Other significant factors concern (3) *dimensionality* [29] (that is, the number of variables in the objective function), (4) *boundary constraints* (that is, the limits imposed on the value of a variable) [30], and (5) *smoothness* of each Gaussian curve (effectively

Table 1. A summary of the ranges selected for the characteristics in our fitness space (F)

Characteristic	**Min**	**Step**	**Max**	**Default**
Number of local optima	0	1	9	3
Ratio of local optima to global optimum	0.1	0.2	0.9	0.5
Dimensionality	1	1	10	2
Boundary constraints	10	10	100	30
Smoothness	10	10	100	15

the gradient) used to generate the landscape [31] - a smaller value indicates a smoother gradient. A summary of the ranges selected for each characteristic is given in Table 1.

3.3 Performance Measurement

In terms of *performance metrics*, we abstract away from algorithm-specific measures, due to the diverse range of methods selected. The following metrics are applied: **(1) Accuracy**: We define this as the mean absolute error of the best solution found on a given set of landscape characteristics, over a number of runs ($\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})$) (where X is the set of best solutions found, n is the number of runs performed and $\bar{x}$ is the known optimum). This is the most commonly-used assessment metric for optimisation algorithms [4]. The generation technique we use creates landscapes with a known global optimum, in this case zero. **(2) Variance of final solutions**: A measure of variation in best solutions found across differently seeded runs. We use the standard deviation of the best solutions of all runs on a given set of landscape characteristics, defined as $(\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2)^{\frac{1}{2}}$ (where X is our data set, n is the size of the data set and $\bar{x}$ is the mean average). **(3) Success rate**: We measure this as the frequency with which differently-seeded runs of an algorithm are able to find a solution within a specified distance from the optimum [32]. We keep the success tolerance relatively low (error less than 1.0×10^{-4}) in order to ensure that we capture the change in success rate of algorithms which perform poorly.

3.4 Experimental Setup

In order to generate the landscapes, we used the Matlab code supplied with [4]. All landscapes were generated using default parameters of three curves, two

dimensions, 0.5 average ratio of local minima to global minimum, 30 units in each dimension with a smoothness coefficient of 15), with only the parameter under investigation changing for each experiment. We ran each algorithm 100 times on each landscape in the set of landscapes generated for each particular characteristic value (when investigating smoothness, for example, we generated 10 different landscapes (smoothness = 10 ... 100), and ran each algorithm 100 times on each landscape).

Parameterisation of algorithms provides a significant challenge when evaluating performance. Our aim is not to perform "competitive testing" [13], but to establish general performance *profiles* for different algorithms over different types of problem. As such, we use the so-called "vanilla" implementation of each algorithm, with general-purpose settings taken from [4]. Where an algorithm has a "population size" parameter, we use a value of 50; where an algorithm has a "range" or "velocity" parameter, we use a value of 10.

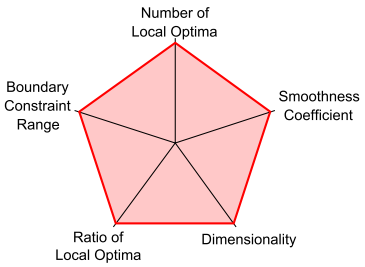
Termination criteria were also standardised. The most objective criterion is the number of objective function evaluations. This means each algorithm has access to the same amount of information from the landscape, and the same amount of *feedback* on potential solutions. Experimentally we determined that the selected algorithms generally converged within 20,000 objective function calculations, so this was used as the termination criterion. The code used for all algorithms, as well as datasets and the landscape generator, is available on request from the authors.

4 Results

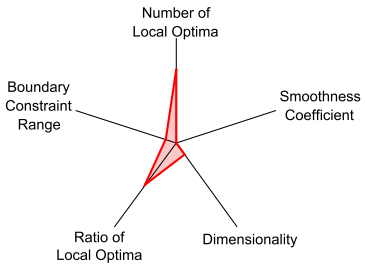
Space prevents a detailed presentation of full experimental plots, but these are available from the project website[1]. To summarise, we plot the *resilience* of each algorithm to changing landscape characteristics, in the form of a radar plot in Figure 1. To assess the resilience of an algorithm we use the standard deviation of the average error across all values of a landscape characteristic, which we normalise on a per-characteristic basis. This "ranking" shows which algorithms do *not* show performance variability versus those which *are* heavily influenced by a characteristic. BFOA shows large deviations in average error for boundary constraint range, smoothness coefficient changes and dimensionality, indicating that BFOA is an algorithm heavily dependent on the landscape of a problem - perhaps because of a heavy reliance on careful parameterisation. SHC also shows large variance - perhaps, in large part again, to a lack of parameters and complicated local optima avoidance techniques. GA and ES show large variation with respect to number of local optima, perhaps supporting the argument that evolutionary algorithms suffer more than most from the problem of becoming "stuck" in local optima.

All algorithms produce the smallest average error when no **local optima** (minima) are present in the fitness landscape. This is expected, as, with only

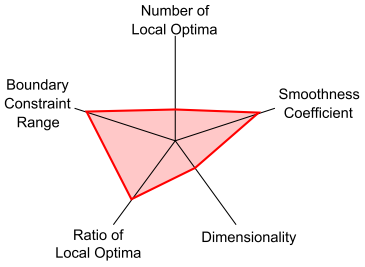
[1] `http://www2.docm.mmu.ac.uk/STAFF/M.Amos/Project/Characterisation`



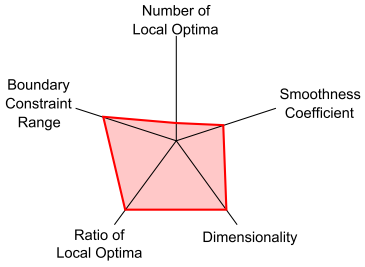
(a) Bees algorithm



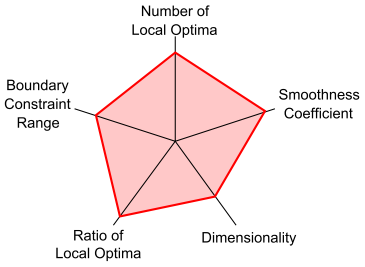
(b) Bacterial foraging optimisation algorithm



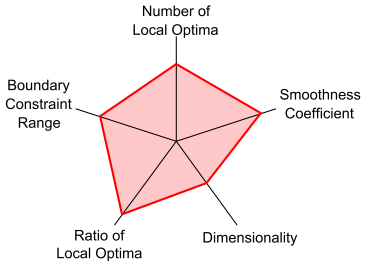
(c) Evolution strategies



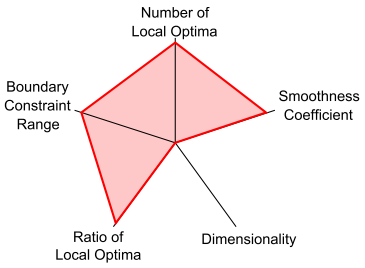
(d) Genetic algorithm



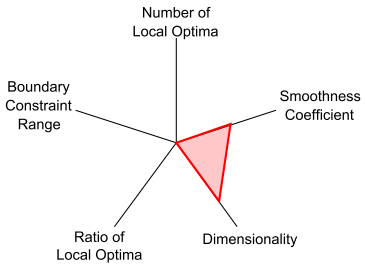
(e) Harmony search



(f) Particle swarm optimisation



(g) Random search



(h) Stochastic hill climbing

Fig. 1. Radar plots depicting the standard deviation of the average error of each algorithm with respect to differing landscape characteristics. Standard deviations are normalised on a per-axis basis. Values close to the centre of the plot indicate a larger variance in average error, indicating these algorithms are more affected by the characteristic. In general, the more robust an algorithm, the larger the plot surface area.

one optimum, there are no alternative solutions to which the algorithms may converge. We observe the greatest average error with only one optimum from SHC, with BFOA (approx. 0.14) also showing a large average error. There are very small average errors (almost zero) from GA, ES, PSO, HS, RS and BA. BFOA also produces the largest variation in final solutions (0.32).With the introduction of only a single local optimum, performance of *most* algorithms degrades significantly. ES and GA suffer significantly, with average error increasing from approximately zero to 0.06 and 0.08 respectively, and the standard deviation of solutions increasing by around 0.15 for each algorithm. SHC also performs poorly, with a similar increase in average error. The least affected are RS (which blindly chooses random solutions, and is therefore unaffected by local minima) and BA, which contains a global search mechanism.

For algorithms which do not directly use the gradient of the landscape, we would expect to see no change in their performance as we adjust the **ratio of local optima** parameter. We observe that RS, which selects new solutions randomly from the entire search space, offers very similar performance in terms of mean error and success rate for all ratio values. Similarly, algorithms which perform a global search should be better at avoiding local minima even when they are attractive - and this is true for BA and HS. PSO shows little change in success rate as the ratio becomes more attractive, owing to the fact that solutions are directed towards the best particle, and their own best solution, regardless of their individual experience with the gradation of the landscape. Interestingly, SHC average error decreases as ratio increases - most likely due to an increased availability of 'better' solutions throughout the landscape. ES demonstrates very poor, yet consistent, performance as the ratio changes. Success rates are very low, and, interestingly, we observe a decrease in the standard deviation of solutions as the ratio increases. This suggests that ES is perhaps more "content" to optimise at a local minima, with the algorithm getting trapped in these more frequently as ratio increases. This could also be true of other algorithms whose deviation decrease, such as BFOA and SHC. GA performs in a similar manner to ES with regard to average error and diversity, although with a considerably better success rate, suggesting that this may be a general problem for algorithms which use an evolutionary approach.

At only one **dimension**, fitness landscapes are trivially easy. The performance of all algorithms reflects this, with all algorithms performing well on landscapes of a single dimension. All algorithms show a success rate (that is, optimisation with an error of under 1.0×10^{-4}) above 90%. As we increase the dimensionality to two, most algorithm performances begin to degrade. Suffering mostly severely is RS, which is to be expected, as random search is our most basic algorithm. Algorithms which also perform poorly at only two dimensions are ES, BA and PSO. It is perhaps surprising, at first, to see BA performing poorly, given that the algorithm contains a randomly sourced global search. However, this global search is *effectively* RS, which performs poorly, so we can assume the global search is not covering enough of the landscape. Coupled with the non-adaptive nature of the algorithm (meaning that solution selection around the current best area is within a relatively large range),

poor algorithm performance is easily explained. We propose that PSO and ES suffer from a similar problem, in that exploration is limited, and neither optimise their current best as accurately as their adaptive variants.

Random search exhibits a similar, yet less extreme, reaction to changes in **boundary constraints** as with the increase in dimensionality. This is to be expected, as the limit on objective function calculations results in random search having less chance to explore the search space. SHC also has an almost linear increase in average error, matching the linear increase in search space size, but produces consistently poor results in terms of success. The social system algorithms (BA and PSO) both exhibit slightly unusual behaviour - as the problem space increases, their success rate also increases. This suggests that their reliance on a parameter to search within a range is hindering the algorithms when the problem space is too small to properly explore. HS provides the best success rate for the entire range of sizes we have selected in this problem, indicating good exploration of the search space irrespective of the range parameter. BFOA also suffers significantly as search space size increases, again implying a heavy reliance on the parameter which controls the range of search for new solutions. The evolutionary algorithms do not cope particularly well with the increase of problem size, with performance in terms of both average error and success rate decreasing consistently as size increases.

The evolutionary algorithms (ES and particularly GA) perform poorly and are most affected by changing the **smoothness coefficient**. BA and PSO all also show decreasing success rate as the curves become steeper, as does BFOA which relies heavily on gradient information. Harmony search suffers similarly to the evolutionary algorithms, and swarm algorithms, as curves become more steep. The similarity in terms of success rate for all algorithms suggests that the availability of gradient information is something which affects all algorithms.

5 Conclusions

In this paper, we have described the results of an extensive study of nature-inspired algorithms, in terms of their performance on fitness landscapes with different characteristics. We studied six nature-based methods (plus two stochastic baseline algorithms), varying a number of landscape features. The most significant characteristic appears to be the number of local minima, where a combination of global and local search appears to be beneficial. On the other hand, the ratio of local optima to the global minimum appears to have little effect on the success of the algorithms under study. As expected, dimensionality proved problematic for all algorithms, whereas landscape smoothness appeared to have little effect.

This work offers a contribution to the empirical study of nature-inspired algorithms, and we hope that it motivates future investigations. To further this work, it may be useful to examine a larger collection of nature-inspired algorithms over a greater range of values for the characteristics, in order to more fully capture a wider variety of algorithmic performance. The current work provides a firm foundation for this.

References

[1] Wolpert, D., Macready, W.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (1997)
[2] Woodward, J.: Why classifying search algorithms is essential. In: 2010 International Conference on Progress in Informatics and Computing (2010)
[3] Morgan, R., Gallagher, M.: When does dependency modelling help? Using a randomized landscape generator to compare algorithms in terms of problem structure. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 94–103. Springer, Heidelberg (2010)
[4] Gallagher, M., Yuan, B.: A general-purpose tunable landscape generator. IEEE Transactions on Evolutionary Computation 10(5), 590–603 (2006)
[5] Rönkkönen, J., Li, X., Kyrki, V., Lampinen, J.: A generator for multimodal test functions with multiple global optima. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 239–248. Springer, Heidelberg (2008)
[6] Jin, Y., Sendhoff, B.: Constructing dynamic optimization test problems using the multi-objective optimization concept. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 525–536. Springer, Heidelberg (2004)
[7] Michalewicz, Z., Deb, K., Schmidt, M.: Test-case generator for nonlinear continuous parameter optimization techniques. IEEE Transactions on Evolutionary Computation 4(3), 197–215 (2000)
[8] Jones, T., Forrest, S.: Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In: Proceedings of the 6th International Conference on Genetic Algorithms, pp. 184–192 (1995)
[9] Chiong, R.: Nature-inspired algorithms for optimisation. Springer (2009)
[10] Zhang, Q.: On the convergence of a class of estimation of distribution algorithms. IEEE Transactions on Computation 8(2), 127–136 (2004)
[11] He, J., Yao, X.: From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. IEEE Transactions on Evolutionary Computation 6(5), 495–511 (2002)
[12] Barr, R., Golden, B., Kelly, J.: Designing and reporting on computational experiments with heuristic methods. Journal of Heuristics 1, 9–32 (1995)
[13] Hooker, J.: Testing heuristics: we have it all wrong. Journal of Heuristics (1995)
[14] Merz, P.: Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Evolutionary Computation 4(4), 337–352 (2000)
[15] Tavares, J., Pereira, F.B., Costa, E.: Multidimensional knapsack problem: a fitness landscape analysis. IEEE Transactions on Systems, Man, and Cybernetics. Part B, Cybernetics 38(3), 604–616 (2008)
[16] Uludag, G., Sima Uyar, A.: Fitness landscape analysis of differential evolution algorithms. In: Fifth International Conference on Soft Computing, Computing with Words and Perceptions in System Analysis, Decision and Control, ICSCCW 2009, pp. 1–4 (2009)
[17] McGeoch, C.: Toward an experimental method for algorithm simulation. INFORMS Journal on Computing 8(1), 1–15 (1996)
[18] Eiben, A.: A critical note on experimental research methodology in EC. In: Proceedings of the 2002 Congress on Evolutionary Computation, pp. 582–587. IEEE (2002)

[19] Brabazon, A., O'Neill, M.: Biologically inspired algorithms for financial modelling. Springer (2006)
[20] Passino, K.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Systems Magazine 22(3), 52–67 (2002)
[21] Pham, D., Ghanbarzadeh, A., Koc, E.: The bees algorithm – a novel tool for complex optimisation problems. In: Pham, D., Eldukhri, E., Soroka, A. (eds.) Intelligent Production Machines and Systems, pp. 454–459 (2006)
[22] Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the Neural Networks, pp. 1942–1948 (1995)
[23] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley (1989)
[24] Bäck, T., Schwefel, H.-P.: An overview of evolutionary algorithms for parameter optimization. Evolutionary Computation 1(1), 1–23 (1993)
[25] Geem, Z., Kim, J.: A new heuristic optimization algorithm: harmony search. Simulation 76(2), 60–68 (2001)
[26] Brownlee, J.: Clever Algorithms: Nature-Inspired Programming Recipes. Lulu (2011), http://www.cleveralgorithms.com
[27] Horn, J., Goldberg, D.: Genetic algorithm difficulty and the modality of fitness landscapes. In: Foundations of Genetic Algorithms, vol. 3 (1994)
[28] Malan, K.M., Engelbrecht, A.P.: Quantifying ruggedness of continuous landscapes using entropy. In: 2009 IEEE Congress on Evolutionary Computation, pp. 1440–1447. IEEE (May 2009)
[29] Hendtlass, T.: Particle swarm optimisation and high dimensional problem spaces. In: 2009 IEEE Congress on Evolutionary Computation, CEC 2009. IEEE (May 1994)
[30] Kukkonen, S., Lampinen, J.: GDE3: The third evolution step of generalized differential evolution. In: 2005 IEEE Congress on Evolutionary Computation, pp. 443–450 (2005)
[31] Beyer, H.-G., Schwefel, H.-P.: Evolution strategies. Natural Computing 1, 3–52 (2002)
[32] Elbeltagi, E., Hegazy, T., Grierson, D.: Comparison among five evolutionary-based optimization algorithms. Advanced Engineering Informatics 19(1), 43–53 (2005)

Evolutionary Generation of Small Oscillating Genetic Networks

Matthijs van Dorp, Bruno Lannoo, and Enrico Carlon

Institute for Theoretical Physics, KULeuven, Celestijnenlaan 200D, 3001 Leuven, Belgium

Abstract. We discuss the implementation and results of an evolutionary algorithm designed to generate oscillating biological networks. In our algorithm we have used a type of fitness function which defines oscillations independent of amplitude and period, which improves results significantly when compared to a simple fitness function which only measures the distance to a predefined target function. We show that with our fitness function, we are able to conduct an analysis of minimal oscillating motifs. We find that there are several different examples of mechanisms that generate oscillations, which make use in various ways of transcriptional regulations, complex formation and catalytic degradation.

Keywords: Genetic algorithms, gene regulatory networks, protein interaction networks.

1 Introduction

Evolutionary algorithms have been used for about 10 years to investigate the structural properties of biological networks [1–6] by constructing them "ab-initio" using a predefined set of evolutionary rules. Here, we use an evolutionary algorithm to generate networks of interacting genes and proteins which have an oscillatory output. The aim is to systematically build up small networks (or "motifs") in order to gain some insights on the core mechanisms responsible for the oscillations. In the generated motifs, protein concentrations follow a stable self-sustained oscillating pattern. The motifs provide examples of oscillators that may be compared to known biological oscillators, such as circadian clocks, and they may improve understanding of the underlying mechanisms of oscillation in this type of networks.

Circadian clocks have long been studied using models that feature a negative feedback loop due to the regulation of gene expression, affecting the production rate of proteins. However, recent results [7] showed that circadian rhythms still persist even when such regulatory mechanisms are disabled. This proves that there are autonomous oscillations due to interactions between proteins, suggesting that the structure of feedback loops is probably more complex than initially expected. Efficient evolutionary algorithms may provide new insights in the modeling of circadian clocks, as many different network architectures could be generated and compared with each other. In addition, experimental techniques are

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 120–129, 2013.

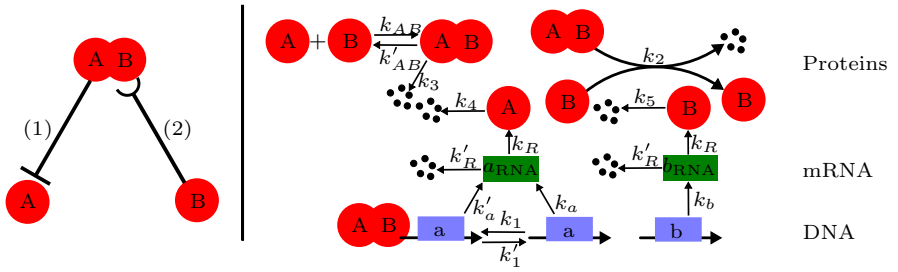


Fig. 1. An example of two representations of the same biological network, with a schematic representation (left) and a complete description with all kinetic rates (right). In the schematic representation, only the "core" interactions (those which define the topology) are given and only proteins are shown. The core interactions shown are (1) a repression of the synthesis of mRNA from gene A by the dimer AB (repression is indicated by a '$\dashv$' symbol, activation by a '$\rightarrow$' symbol), and (2) a catalytic degradation of the dimer AB by B, i.e. the reaction $B + AB \rightarrow B$ which is indicated as '—$\subset$'. The groups of small dots in the full representation indicate degradation.

rapidly improving, and in the near future high quality data is expected to allow discrimination between motifs based on their specific output.

In this paper we revisit an algorithm, originally proposed in [1], by improving its efficiency and extending it further. We show that such an algorithm is capable of rapidly generating oscillating motifs thanks to optimally selected score functions. We show how the different types of biomolecular reactions may be combined to produce an oscillating output. Quite interestingly, in view of the recent experimental results [7], the output also provides examples of purely post-transcriptional oscillators which do not rely on oscillations of mRNA concentrations.

2 Algorithm

The allowed biological mechanisms and interactions used in this work are transcriptional regulation, formation and dissociation of protein complexes, and catalytic degradation. They are shown graphically in Figure 1(right), which provides an example of a small network in its full representation. In Fig. 1(left) we provide a schematic representation (as used throughout the paper) of the same network where only proteins are shown. The latter representation omits the genes, which produce mRNA, and the mRNAs which produce the gene-specific protein. In order to simulate the temporal behavior of a network, we use deterministic mass-action kinetics. The resulting set of nonlinear first-order ODEs are solved by the Runge-Kutta-Fehlberg Method (RKF45).

2.1 Evolutionary Fitness

An appropriate fitness function is highly important both for fast convergence of the algorithm and for the evolution of motifs that are as small as possible.

Additionally, the choice of the fitness function will cause certain evolutionary pathways to be much more accessible than others, and therefore the results will depend strongly on the fitness function that is chosen. Previous algorithms designed for the same purpose have varied the exact selection mechanism, such as elitist evolution or tournament selection [8, 9], but the notion of fitness has always been related to a specific concentration profile. We have abandoned this kind of scoring, as the associated requirements for amplitude and period are very restrictive. Instead, a fitness function should reward oscillatory behavior without being tied to any predefined shape, and not even distinguish between sawtooth profiles, pulses, and sine-like waves. The most straightforward way of defining oscillation independent of the profile is to analyse its peak-to-trough behavior. This leads to the fitness function

$$S = 20 - 2\sum_{i=1}^{10} \frac{|a_i - a_{i+1}|}{a_i + a_{i+1}} \min\left(1, |a_i - a_{i+1}|\right), \tag{1}$$

where the a_i are the first 11 extrema, ordered by time, of the concentration of a target protein $A(t)$. If $A(t)$ has less than 11 extrema, the sum only includes the available terms, and $S = 20$ for all monotonic functions. Equation (1) is the mathematical equivalent of the intuitive concept that oscillations are of a good quality if the peaks are relatively much higher than the troughs. The $\min\left(1, |a_i - a_{i+1}|\right)$ on the right is a technical correction which we added to prevent the algorithm from evolving very low-concentration oscillations. A significant advantage of this specific scoring function is that a strongly damped oscillation is recognised and rewarded, which is helpful for fast convergence. The convergence threshold is set to $S = 4$ since for such values stable oscillations are seen in any network, except for some rare cases which feature very weakly damped oscillations.

2.2 Topological Reduction

In spite of choosing a fitness function that favors fast convergence, the evolutionary procedure almost inevitably enlarges the size of the network by adding links and nodes which are not necessary for oscillatory behavior. The simplest topological reduction would consist of cutting away parts of the network as long as this does not destroy oscillations. However, it is likely that the network can be reduced further if the kinetic rates are varied to compensate for the removal of a component. Therefore, we have applied additional topological evolution, which computes a topological fitness score related to the topological components. We constrained the evolution by removing any networks that did not satisfy $S < 5$ in order to preserve oscillatory function, but other than this constraint the topological evolution could in principle evolve freely towards smaller networks. In order to improve convergence, we also incorporated the kinetic rates into the topological fitness score such that interactions could have their importance reduced gradually, increasing the likeliness that they could be removed entirely in a future generation.

2.3 Evolutionary Process

In order to illustrate the evolutionary algorithm, let us consider the pseudocode for the functions "evolve()" and "reduce_topological_size()" as displayed below.

```
function evolve()
  make_initial_network()
  for n = 0 to 300
    mutate_networks()        /* Create mutant for each network */
    score_networks()         /* Compute S for all networks */
    if S_min ≤ 4
      break                  /* Stop evolution once fitness threshold is reached */
    prune_networks(100)      /* Keep only 100 best scoring networks */
    if T > 15
      reduce_topological_size(S_min + 1)
  if S_min > 4
    return                   /* Failed to evolve oscillations */
  reduce_topological_size(5) /* Reduce T of oscillating network */
  for n = 0 to 20            /* Optimize network score */
    mutate_networks_kinetic()  /* Only evolve rate constants */
    score_networks()
    prune_networks(100)
  save_best_network()        /* Store network with lowest S */
```

```
function reduce_topological_size(int maxscore)
  for n = 0 to 500
    mutate_networks()        /* Create mutant for each network */
    check_fitness(maxscore)  /* Remove networks with bad fitness */
    score_topology()         /* Compute T for all networks */
    prune_networks(10)       /* Keep only 10 best scoring networks */
```

The first function, evolve(), shows the general course of evolution. Every generation consists of duplicating all networks and mutating each duplicate in a random way. These mutations include topological modifications as well as changes to kinetic rate constants. Subsequently, the score of all networks is computed and the worst performing networks are discarded. The second function, as described in Sec. 2.2, performs a topological reduction which aims to shrink the network size, and it is used by the first function when the networks have grown too large, or in order to reduce the size of an evolved network with oscillatory behavior.

3 Results

The evolution results in a large pool of networks of various complexities. As the number of possible topologies is small when the complexity of the graph is restricted, we analyse the collection of about 15,000 evolved networks with a topological size $T \leq 5$ (where T is defined as the total number of genes,

protein complexes, regulations, and catalytic degradations). The total number of distinct topologies was less than 100. We shall discuss the differences and similarities between the topologies obtained.

3.1 Parameter Variation

The population of oscillating networks that results from repeatedly running an evolutionary algorithm depends not only on the fitness function, which is the same for all networks discussed here, but also on the choice of parameters and evolutionary limits. A very broad range of allowed kinetic rates decreases the performance of the algorithm, in particular because it increases the computational cost of numerically solving the system of ODEs for a network. We use a fixed time window within which the oscillations should take place, which is sufficiently large compared to allowed kinetic rates such that a wide range of periods is found (across more than one order of magnitude). Effectively, the minimum possible periods are dictated by the upper allowed limits of kinetic rates, while the maximum possible period is a consequence of the fitness computation which demands a minimum number of oscillations in the maximum time allowed for integration. We use several parameter settings in order to search different regions of parameter space. In the default setting, we assume kinetic rates to be around 1.0 such that all timescales in the system are similar. In a second case we assume that the formation of complexes is significantly faster than transcriptional regulation and degradation. In a third setting we have disabled the evolution of catalytic degradations, which lead to a larger proportion of repressing regulations, and in a final setting we loosen the limits on all mutable rates such that the search space is significantly larger.

3.2 Mechanisms of Oscillation

Oscillatory function is known to be connected to negative feedbacks that incorporate a time delay, and our simplest networks use no more than a few distinct patterns in spite of belonging to many different topologies. Examples of those patterns are given in Figures 2, 3, 4 and 5.

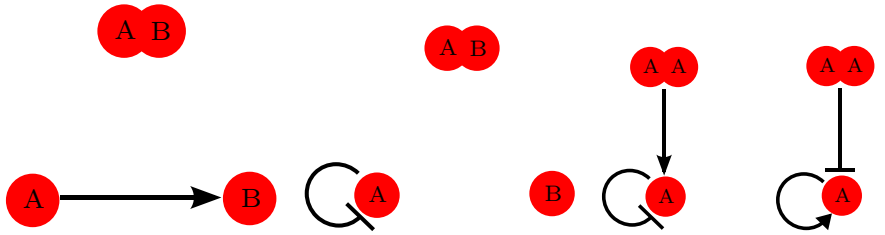
(i) Motifs without catalytic degradations. A few examples are shown in Fig.2 and 3. These networks fall into two subcategories, namely networks with two proteins and one regulation, and networks with one protein and two regulations. Both types feature multimers which help to generate time delays, and in most cases the complex formation is irreversible. Of note, there are two types of time delays due to complexation, namely competitive complexation, where two different ways of complex formation compete for the same protein (e.g. Fig. 3(a)), and chain-like complexation, where complexes of more than two components are formed (e.g. Fig. 3(b)). Oscillating networks with $T = 4$ (see Fig. 2) all belong to this group.

(ii) Motifs with catalytic degradations and regulations, such as the examples in Fig. 4. Again there exist two subcategories, because mechanisms can have

either two proteins and a single heterodimer formed by the two proteins; or only one protein, a dimer and either a trimer or a tetramer.

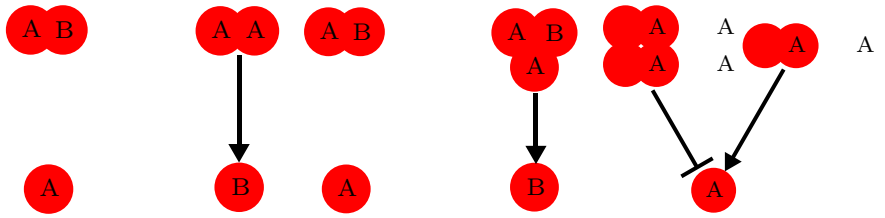
(iii) Motifs without regulations. They can also have one or two genes, and all feature reversible complexation as a method of generating a time delay. Examples are shown in Fig. 5.

(iv) Motifs without multimers. They all have two genes and are similar to certain motifs from (i) where the absence of the AB heterodimer is compensated for by catalytic degradation interactions. Because of the similarity, no examples are shown.



(a) The mixed feedback loop (MFL) (b) The mixed autorepression loop (MAL) (c) and (d) Two dimer-monomer (DMO) oscillators

Fig. 2. All four evolved motifs of size $T = 4$. The MFL and MAL motifs show a negative feedback by a regulation whose action is delayed due to the formation of the AB dimer, which acts as an exclusion mechanism. The MFL motif has been previously studied in [10]. The DMO motifs employ a repressing regulation as a negative feedback, while the time delay is due to TF binding/unbinding dynamics rather than dimerization.



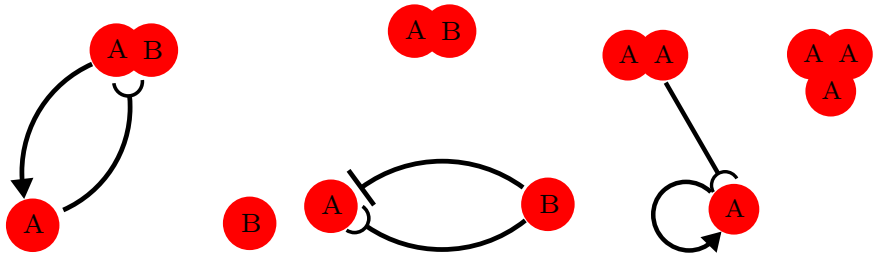
(a) Time delay due to competition (b) Time delay due to high-order complex (c) Single protein oscillator

Fig. 3. Three examples of pattern (i). Most motifs with this pattern are similar the MFL, MAL and MDO motifs presented in Fig. 2. Time delays are larger than in the original motifs due to usage of higher order multimers, such that oscillations are evolved more easily.

4 Algorithm Performance

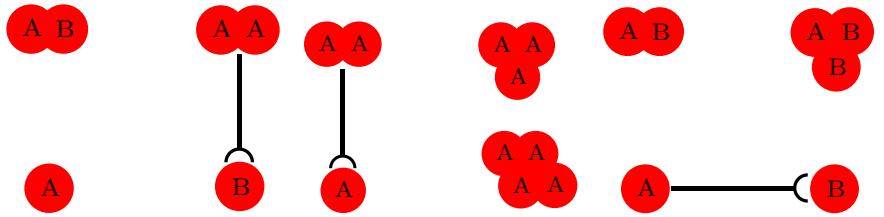
4.1 Efficiency

We have tried to optimize our algorithm towards generating networks at the fastest possible rate. We terminated our evolution after 300 generations, as the



(a) Negative feedback by degradation of activator (b) Negative feedback only due to reversible complexation (c) Delay of self-degradation by reversible trimer formation

Fig. 4. Examples of pattern (ii). Negative feedbacks due to catalytic degradations are often accompanied by reversible formation of a complex (denoted by gray dashed arrows), which results in a time delay.



(a) Time delay by competition and dimer (b) Time delay due to formation of high-order multimers (c) Time delay due to double chain formation

Fig. 5. Examples of pattern (iii). When no regulations are present, the catalytic degradation seems to need two separate time delay mechanisms in order to cause stable oscillations. This is seen as either two levels of reversible complexation as seen in (b) and (c), or as a reversible complexation and a dimer.

evolutionary process had likely become stuck in a local minimum if still no oscillations had been found. This was not a frequent problem, however, and depending on the parameter setting the success rate varied from 75% to 95%. The typical type required to complete one evolutionary process varied (depending on the parameter settings) from a few minutes to half an hour when running the program on three cores of a standard quad-core personal computer. In exceptional cases the computation time can be much longer. In addition, we enforced a maximal number of time integration steps since our RKF45 integration routine was very inefficient in some cases. If this number of steps was exceeded, the network was removed from the population regardless of its fitness performance.

In Fig. 6 we have shown how the topological size T of the network depends on the number of evolutionary steps. In line with our expectations, networks typically end up smaller when evolution finishes early. More generations do not only increase the size of the networks by mutations, but they also integrate existing parts of the network better and better, such that it becomes harder to prune these parts once oscillations have been evolved.

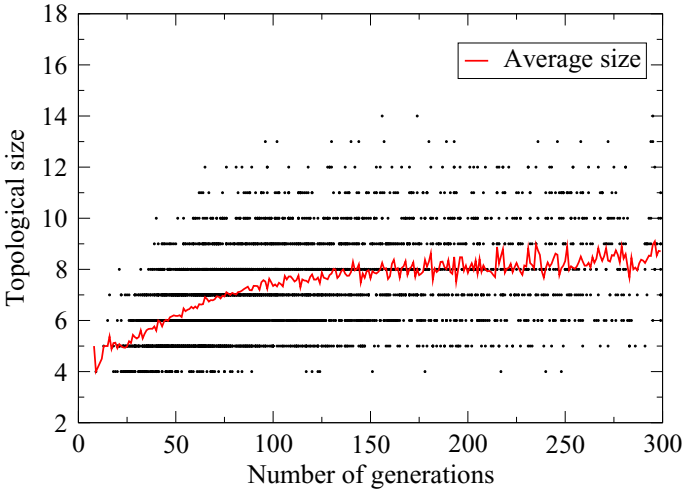


Fig. 6. The typical size of the final, reduced network grows with the number of generations it takes to evolve oscillations. The average was computed for all 39, 457 networks evolved in the large-range setting, and a random subset of 3917 networks are represented by dots. If no oscillations had evolved after 300 generations, the evolutionary process was abandoned.

4.2 Influence of Parameter Choices

As discussed in Sec. 3.1, we used four different parameter settings. The reason to do this was twofold: firstly, to avoid generating only networks in a very limited region of parameter space, and secondly, to allow for a comparison of results, such that we can estimate to what degree the results are a consequence of the parameters of the algorithm.

It turns out that some parameter choices exclude a significant portion of possible motifs. The default choice allows catalytic degradation, but except in rare cases, it failed to find small networks incorporating catalytic degradations with size $T \leq 5$. When the catalytic degradation was removed as an evolutionary option, the rate of finding networks with $T = 5$ increased from 3.3% to 5.2% and several mechanisms featuring repressing regulations became much more common. The setting with fast complex formation yielded several networks with $T = 4$, but the variety of networks was limited to only the MFL and MAL networks shown in Figures 2(a) and 2(b). The most successful setting was the default setting with very loose parameter limits, where the variable rates were allowed to vary across four orders of magnitude instead of just two. Nevertheless, this setting appeared to be inefficient for the evolution of networks with repressing regulations, such that some motifs found in other settings were not evolved, and it seems plausible that a parameter setting can be found which is much more conducive towards evolving motifs featuring repressing regulations. Additionally, this setting consumed roughly ten times more computation time per evolutionary

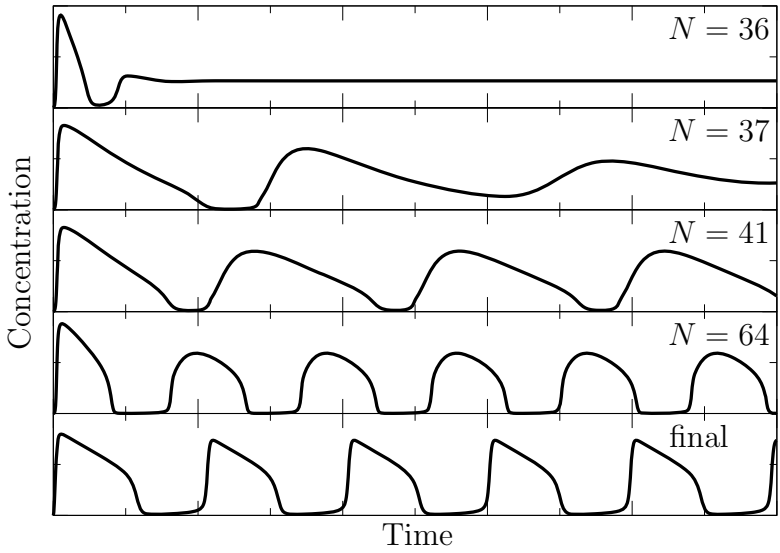


Fig. 7. An example of the evolution of sustained oscillations for a network with a topology as shown in 5(b). The amplitude, period and shape of the oscillations vary while the quality of the oscillation is gradually improving. Each graph shown represents the output for the best-scoring network in the N-th generation, where the values of N were chosen from a single run. The final graph shows the output of the final network after topological reduction.

process than the other settings. Broadly speaking, it seems likely that there is no single setting which is optimal, and a comprehensive set of oscillating motifs can only be generated by varying the parameter settings.

5 Conclusions

In this paper, we have presented an analysis of various examples of oscillatory biological networks generated by an evolutionary algorithm. The algorithm can efficiently produce oscillating networks of small size. We have discussed which choices in the implementation of the evolutionary algorithm were responsible for this improvement, and how our implementation in general influences what motifs are evolved.

The most significant improvement resulted from a fitness function that did not force evolution towards any particular type of oscillation. The freedom that resulted from a loose definition of the concept of oscillations allowed for the evolution of many small networks. Additionally, the algorithm was made more efficient by avoiding the evolution of very large networks.

Many different types of motifs appear to have an oscillatory output for certain rate constants. This is a positive feature of the algorithm since it suggests that it performs a wide search in the complex space of all possible topologies. In view

of the recent experiments on circadian clocks [7], it is also interesting that the algorithm generated several examples of post-transcriptional oscillators.

The abundance of motifs found by our algorithm might prove useful, serving as a set of candidates that might be found as core parts of networks found in systems biology. The small systems we have found are likely simple compared to biological mechanisms. However, as they hint at topological structures that are conducive towards oscillatory behavior, they may help in recognizing the core parts of biological examples of oscillating protein networks.

In this paper we have only evolved networks aimed at oscillatory output. Real biological evolution may have further selected through the available oscillating motifs using criteria as robustness to fluctuations and/or entrainability (i.e. the resetting of the phase of the oscillations). Further research aimed at analyzing the properties of the generated motifs against these types of criteria may suggest which motifs are most likely to be present in real biological networks.

References

1. François, P., Hakim, V.: Design of genetic networks with specified functions by evolution in silico. Proc. Nat. Acad. Sci. USA 101(2), 580–585 (2004)
2. Chu, D.: Evolving genetic regulatory networks for systems biology. In: Proc. IEEE Congress Evolutionary Computation, CEC 2007, pp. 875–882 (2007)
3. Sakamoto, E., Iba, H.: Inferring a system of differential equations for a gene regulatory network by using genetic programming. In: Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1, pp. 720–726 (2001)
4. Floares, A.: Automatic inferring drug gene regulatory networks with missing information using neural networks and genetic programming. In: IEEE International Joint Conference on Neural Networks, pp. 3078–3085 (2008)
5. Tagkopoulos, I., Liu, Y., Tavazoie, S.: Predictive behavior within microbial genetic networks. Science 320(5881), 1313 (2008)
6. Šter, B., Avbelj, M., Jerala, R., Dobnikar, A.: On the Origin and Features of an Evolved Boolean Model for Subcellular Signal Transduction Systems. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 383–392. Springer, Heidelberg (2011)
7. O'Neill, J.S., van Ooijen, G., Dixon, L.E., Troein, C., Corellou, F., Bouget, F.Y., Reddy, A.B., Millar, A.J.: Circadian rhythms persist without transcription in a eukaryote. Nature 469, 554–558 (2011)
8. Paladugu, S.R., Chickarmane, V., Deckard, A., Frumkin, J.P., McCormack, M., Sauro, H.M.: In silico evolution of functional modules in biochemical networks. IEE Proceedings Systems Biology 153(4), 223–235 (2006)
9. Jin, Y., Meng, Y., Sendhoff, B.: Influence of regulation logic on the easiness of evolving sustained oscillation for gene regulatory networks. In: Proc. IEEE Symp. Artificial Life ALife 2009, pp. 61–68 (2009)
10. François, P., Hakim, V.: Core genetic module: the mixed feedback loop. Phys. Rev. E 72, 031908 (2005)

Using Scout Particles to Improve a Predator-Prey Optimizer

Arlindo Silva[1], Ana Neves[1], and Teresa Gonçalves[2]

[1] Escola Superior de Tecnologia do Instituto Politécnico de Castelo Branco
{arlindo,dorian}@ipcb.pt
[2] Universidade de Évora
tcg@uevora.pt

Abstract. We discuss the use of scout particles, or scouts, to improve the performance of a new heterogeneous particle swarm optimization algorithm, called scouting predator-prey optimizer. Scout particles are proposed as a straightforward way of introducing new exploratory behaviors into the swarm, expending minimal extra resources and without performing global modifications to the algorithm. Scouts are used both as general mechanisms to globally improve the algorithm and also as a simple approach to taylor an algorithm to a problem by embodying specific knowledge. The role of each particle and the performance of the global algorithm is tested over a set of 10 benchmark functions and against two state-of-the-art evolutionary optimizers. The experimental results suggest that, with the addition of scout particles, the new optimizer can be competitive and even superior to the other algorithms, both in terms of performance and robustness.

Keywords: swarm intelligence, particle swarm optimization, heterogeneous particle swarms.

1 Introduction

The particle swarm optimization algorithm (PSO) is a stochastic optimization algorithm that uses a population of individuals, represented as vectors of real numbers, to search for the global optimum in a multidimensional space [7]. Individuals are usually called particles, and the particle set is called a swarm. In addition to their current position, each individual keeps a memory of the best position it has found so far, as well as a velocity vector. Each particle is also aware of the best position found by its best neighbor in the swarm. The population behavior mimics the social interactions in real swarms by computing each particles' velocity in terms of how strongly the individual is attracted to its own notion of where the best solution should be and the believe of the group, represented by the best neighbor. The velocity vector is added to the particle's position at each iteration, thus defining its trajectory in the search space.

Qualities like conceptual simplicity, quick implementation, low computational costs and being easily adaptable to new domains have made the PSO hugely

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 130–139, 2013.

popular amongst practitioners, with successful applications in many areas [11]. Despite its popularity, the basic PSO also presents some challenges that must be tackled for a successful application [1,14]. These include controlling the balance between exploration and exploitation; maintaining some level of diversity in the swarm after it has converged (there is no equivalent to the mutation operator present in evolutionary algorithms); fine-tuning a reasonable (but not optimal) solution to where the swarm has converged; avoiding performance degradation when optimizing non-separable functions.

Since the introduction of the original PSO, there has been a very strong body of research dedicated to the study and overcome of the algorithm perceived weaknesses [12]. Amongst the many variants in use today, we can identify three main groups of approaches that seem the most promising: hybrid approaches, which use one or more mutation operators mainly inspired in other evolutionary algorithms [6]; memetic variants, which ally the advantages of local search algorithms with the global exploratory capabilities of the PSO [10] and, more recently, heterogeneous particle swarm optimizers, where different particles within the same swarm can have different behaviors and/or properties, allowing for different parts of the swarm to be tuned for different aspects of the problem being optimized or for different phases of the exploration process [4,8].

In previous work we introduced a new heterogeneous particle swarm algorithm, called scouting predator-prey optimizer (SPPO), which uses an extra particle, called a predator, to dynamically control diversity and the balance between exploitation and exploration [16]. The same article introduced the concept of scout particles, which are a swarm subset that can be updated with different rules, leading to alternative exploratory behaviors. In this paper, we discuss the effects of different scout particles, and show how they can perform different roles, from globally improving the SPPO performance, to using problem specific knowledge to better adapt the algorithm to a specific task.

2 The Scouting Predator-Prey Optimizer

In particle swarm optimization each swarm member is represented by three m-size vectors, assuming an optimization problem $f(\mathbf{x})$ in $\mathbb{R}^m$. For each particle i we have a $\mathbf{x}_i$ vector that represents the current position in the search space, a $\mathbf{p}_i$ vector storing the best position found so far and a third vector $\mathbf{v}_i$ corresponding to the particle's velocity. For each iteration t of the algorithm, the current position $\mathbf{x}_i$ of every particle i is evaluated by computing $f(\mathbf{x}_i)$. Assuming a minimization problem, $\mathbf{x}_i$ is saved in $\mathbf{p}_i$ if $f(\mathbf{x}_i) < f(\mathbf{p}_i)$, i.e. if $\mathbf{x}_i$ is the best solution found by the particle so far. The velocity vector $\mathbf{v}_i$ is then computed with equation 1 and used to update the particle's position (equation 2).

$$\mathbf{v}_i^{t+1} = w\mathbf{v}_i^t + \mathbf{u}(0, \phi_1) \otimes (\mathbf{p}_i - \mathbf{x}_i^t) + \mathbf{u}(0, \phi_2) \otimes (\mathbf{p}_g - \mathbf{x}_i^t) \quad (1)$$

$$\mathbf{x}_i^{t+1} = \mathbf{x}_i^t + \mathbf{v}_i^t \quad (2)$$

In equation 1 $(\mathbf{p}_i - \mathbf{x}_i^t)$ represents the distance between a particle and its best position in previous iterations and $(\mathbf{p}_g - \mathbf{x}_i^t)$ represents the distance between a particle and the best position found by the particles in its neighborhood, stored in $\mathbf{p}_g$. $\mathbf{u}(0, \phi_1)$ and $\mathbf{u}(0, \phi_2)$ are random number vectors with uniform distributions in intervals $[0, \phi_1]$ and $[0, \phi_2]$, respectively. w is a weight that decreases linearly with t and $\otimes$ is a vector component-wise multiplication.

2.1 The Predator Effect

One of the limitations of the standard particle swarm algorithm is its inability to introduce diversity in the swarm after it has converged to a local optimum. Since there is no mechanism similar to a mutation operator, and changes in $\mathbf{x}_i$ are dependent on differences between the particles' positions, as the swarm clusters around a promising area in the search space, so does velocity decreases and particles converge. This is the desirable behavior if the optimum is global, but, if it is local, there is no way to increase velocities again, allowing the swarm to escape to a new optimum, i.e. there is no way to go back to a global exploration phase of search after the algorithm entered a local search (exploitation) phase.

We introduced the predator-prey idea to alleviate this problem [15,16]. The predator particle's velocity $\mathbf{v}_p$ is updated using equation 3, oscillating between the best particle's best and current position. This update rule makes the predator effectively chase the best particle in the search space.

$$\mathbf{v}_p^{t+1} = w\mathbf{v}_p^t + \mathbf{u}(0, \phi_1) \otimes (\mathbf{x}_g^t - \mathbf{x}_p^t) + \mathbf{u}(0, \phi_2) \otimes (\mathbf{p}_g - \mathbf{x}_p^t) \tag{3}$$

The role of the predator particle in the SPPO algorithm is to introduce a perturbation factor in the swarm and to guarantee that this disturbance increases as the swarm converges to a single point. To achieve this, we add a perturbation to a particles's velocity in dimension j, as defined by equation 4, where $u(-1, 1)$ and $u(0, 1)$ are random numbers uniformly distributed between the arguments, x_{max} and x_{min} are, respectively the upper and lower limit to the search space and r is the user defined perturbation probability.

$$v_{ij}^t = v_{ij}^t + u(-1, 1)|x_{max} - x_{min}|, \text{if } u(0, 1) < r \exp^{-|x_{ij} - x_{pj}|} \tag{4}$$

From equation 4 follows that a random perturbation is added to the velocity value in dimension j with a probability that depends on the particles's distance to the predator in that dimension. This probability is maximum (r) when that distance is 0 but rapidly decreases if the particle escapes the predator. Since the predator chases the best particle, the perturbation in the swarm is more likely when all the particles are very near, i.e. during the exploitation phase, and becomes almost inexistent when the particles are far apart. This mechanism allows for particles to escape and find new optima far from the current attractor even in the last phases of exploitation.

2.2 Scout Particles

The predator-prey optimizer is an heterogeneous particle swarm algorithm, since the predator particle is updated using a different equation. We propose a new level of heterogeneity by including scout particles into the swarm. Scout particles, or scouts, are a subset of the swarm that implements different exploration strategies. They are inspired by the common presence in real insect swarms, e.g. honeybees [2], of specific individuals in charge of scouting for new food sources. When successful in this task, they communicate the finding to other individuals, called recruits, that may then follow to the new food source.

In our algorithm, scouts are specific particles to which we assign update rules different form those used by the rest of the swarm. Each new rule implements a different exploratory behavior with specific objectives in terms of function optimization. As in real swarms, resource management is essential, so we must ensure that the positive effect of the scouts is not offset by many extra function evaluations or memory requirements. It is also important to ensure that the addition of scouts doesn't disrupt the swarm usual behavior. To achieve this, recruitment is only done by updating the scout best found position, as well as its neighborhood best when that is the case. From the viewpoint of the rest of the swarm the use of scouts is, consequently, transparent, constituting a very flexible way to introduce new capabilities to the optimization algorithm, or even to hybridize it with a different optimizer altogether.

Scouts can be used to introduce improvements to the global algorithm, e.g. a local search sub-algorithm, or to implement problem dependent mechanisms to better adapt the algorithm to a specific problem. In the later case, the mechanism can be of general nature or based in specific problem knowledge that may be used to speed up the optimization process. To illustrate these ideas, we will use two scout particles to improve the performance of the predator-prey algorithm in continuous optimization problems and a third particle to taylor the algorithm to a purposefully designed problem set.

For the first scout we choose the best particle in the swarm at a given iteration and perform a random mutation on one of its dimensions j using equation 5, where $n(0, \sigma^2)$ is a a random number drawn from a normal distribution with average 0 and standard deviation σ. $\mathbf{p}_g$ is updated to the new $\mathbf{p}'_g$ only if $f(\mathbf{p}'_g) < f(\mathbf{p}_g)$. σ starts at $x_{max}/10$ and is updated during the run using the $1/5$ rule borrowed from evolution strategies [3]: after every 100 generations σ is doubled if the mutation success rate is over $1/5$ and is halved otherwise. This mutation mechanism allows for a local search to be made around $\mathbf{p}_g$ over time.

$$p'_{gj} = p_{gj} + n(0, \sigma) \tag{5}$$

The second scout particle uses an update rule inspired by opposition based learning (OBL) [18]. The basic idea behind OBL is that sometimes it might be useful to search in the opposite direction of the current position. Opposition based extensions of particle swarm [19] and differential evolution [13] have improved on the results of the corresponding base algorithms. For this second scout, we use the particle with worst evaluation, $\mathbf{p}_w$, based on the heuristic that in the

opposite of the worst particle might be a more promising search region. We compute the opposite particle position $\mathbf{p}'_w$ using equation 6 for each dimension j, with a_j^t and b_j^t being, respectively, the maximum and minimum value for p_{ij} at generation t. Again, $\mathbf{p}_w$ is only updated to the new $\mathbf{p}'_w$ if $f(\mathbf{p}'_w) < f(\mathbf{p}_w)$.

$$p'_{wj} = a_j^t + b_j^t - p_{wj} \tag{6}$$

Scout particles are updated prior to the main update cycle of the swarm, where they can cumulatively be updated using equations 1 and 4. To save objective function evaluations, we update the best particle but not the worst one. The third scout particle is problem dependent and it will be described in the experimental results section, together with the relevant experiments.

3 Experimental Results

For the experiments described in this paper we used 10 common benchmark functions, selected from the optimization bibliography. The first three are unimodal functions, but f_2 is non-separable and in f_3 the optimum is in a very flat zone with little information to guide the algorithms. f_4-f_{10} are multimodal functions with many local optima, selected to pose different obstacles to the optimization algorithms. Table 1 lists names and parameters for the benchmark functions, including the optimum value $f(\mathbf{x}^*)$ and its position x_i^*.

Table 1. Benchmark functions' names and parameters

Function	Name	Range	$f(\mathbf{x}^*)$	x_i^*	Displacement
f_1	Sphere	$[-100 : 100]$	0	0	25
f_2	Rotated Ellipsoid	$[-100 : 100]$	0	0	25
f_3	Zhakarov's	$[-5 : 10]$	0	0	1.25
f_4	Rastrigin's	$[-5.12 : 5.12]$	0	0	1.28
f_5	Ackley's	$[-32 : 32]$	0	0	8
f_6	Griewangk's	$[-600 : 600]$	0	0	150
f_7	Salomon's	$[-100 : 100]$	0	0	25
f_8	Kursawe's	$[-1000 : 1000]$	0	0	250
f_9	Shaffer's	$[-100 : 100]$	0	0	25
f_{10}	Levy-Montalvo's	$[-500 : 500]$	0	1	125

From Table 1 we can observe that all used functions (f_{10} being the exception) have their optimum in the search space origin. They all also present some level of symmetry around that point. While these characteristics will later be useful to set up some specific experiments, they could also give some advantage to algorithms that present a bias towards the origin of the search space. To avoid this, the global optimum was displaced, for each function, by the amount shown on Table 1. All functions were optimized in 40 dimensions in the reported experiments and their equations can be found in [16] as well as in the field bibliography.

The first set of experimental results compares the final SPPO algorithm, using both local search and an opposition based search scouts, against other PSO

and differential evolution (DE) based approaches [17]. We included DE algorithms since differential evolution shares many of the PSO advantages, namely in terms of simplicity, flexibility and performance. We used a recent state-of-the art hybrid PSO (HMRPSO) variant [5], which reported very good results in the optimization of a large set of benchmark functions when compared with several other PSO variants and evolutionary algorithms.

The DE approach used for comparison, which also showed promising experimental results, is called free search differential evolution (FSDE) [9]. Our implementations were first tested on the benchmarks reported in the original papers to minimize implementation discrepancies. Standard versions of the PSO and DE algorithms were also included in our experiments.

The parameters for the algorithms are the ones proposed by the respective authors. For the scouting predator-prey optimizer we used $\phi_1 = \phi_2 = 1.6$, w was decreased from 0.4 to 0.2 and $r = 0.0008$. Population size was set to 20 and the algorithms were run for $1e4$ iterations or until $2e5$ objective function evaluations were performed. In Table 2 we present averages and standard deviations for the best values found for each test function, taken over 50 runs of each pair algorithm/function. The random number generator was initiated to ensure that all algorithms started with the same population for corresponding runs.

Table 2. Experimental results for the final algorithm: average and standard deviations of the best values found over 50 runs of an algorithm for each function

	PSO	DE	HMRPSO	FSDE	SPPO
f_1	1.65169e-26 (5.54796e-26)	6.25047e-23 (4.4143e-22)	3.38079e-23 (1.4217e-23)	0.242096 (0.174053)	**7.57306e-31** **(3.02794e-30)**
f_2	316.126 (254.974)	56.2572 (397.747)	0.00226426 (0.00212582)	61.5534 (33.1706)	**2.52261e-09** **(2.40497e-09)**
f_3	15.8846 (9.53778)	0.000692142 (0.0016271)	5.19511e-09 (4.86953e-09)	1.13648 (0.807662)	**6.72887e-10** **(5.23577e-10)**
f_4	55.8172 (17.0791)	148.852 (84.0428)	17.8104 (5.6274)	7.72839 (11.3974)	**0.0198992** **(0.140708)**
f_5	0.476326 (2.91565)	5.47737 (6.9252)	1.27917e-12 (2.26036e-13)	0.150284 (0.0844456)	**3.96305e-14** **(8.27882e-15)**
f_6	0.0134364 (0.0169437)	0.0861482 (0.282405)	**0.00777161** **(0.0112705)**	0.290238 (0.191083)	0.0524945 (0.0498488)
f_7	0.639873 (0.137024)	0.389874 (0.194044)	0.553874 (0.0973316)	**0.375873** **(0.169706)**	1.19387 (0.219842)
f_8	-41.2553 (50.7622)	-76.2049 (67.5841)	-24.9186 (25.9715)	171.819 (215.74)	**-150.326** **(16.6553)**
f_9	5.15659 (2.34525)	15.976 (0.391461)	6.77497 (1.32032)	12.0213 (1.81459)	**0.860774** **(0.424196)**
f_{10}	0.0181309 (0.126894)	0.0621908 (0.216245)	5.56247e-22 (3.83363e-22)	55.1298 (78.7154)	**5.86288e-28** **(3.38157e-28)**

As we can see in Table 2, the SPPO algorithm obtained substantially better average results in 8 of the 10 benchmark problems. For the remaining two functions, the best result was achieved by the HMRPSO for f_6 and by $FSDE$ for f_7.

All the hybrid algorithms performed consistently better than the basic versions, which can be observed in the columns marked PSO and DE in Table 2.

These results confirm more extensive results presented in previous work [16], where, using a significantly larger problem set, we could also observe that the scouting predator-prey optimizer was generally competitive and frequently superior to other state-of-the-art PSO and DE based algorithms. In this work we aim to additionally address the different roles performed by each scout particle and also to illustrate how scout particles can be used to improve performance by using problem specific knowledge.

The second set of experiments compares, under the same experimental conditions used previously, the standard PSO, the PSO with just the predator effect added (PPO), the PPO with an opposition learning based scout (OPPO), the PPO with a local search based scout (LPPO) and the final algorithm including all described behaviors, the SPPO. The results are presented in Table 3.

Table 3. Experimental results for the general scout particles: average and standard deviations of the best values found over 50 runs of an algorithm for each function

	PSO	PPO	OPPO	LPPO	SPPO
f_1	1.65169e-26 (5.54796e-26)	**1.71656e-28** **(3.5187e-28)**	1.63629e-27 (5.77022e-27)	**0** **(0)**	**7.57306e-31** **(3.02794e-30)**
f_2	316.126 (254.974)	1260.18 (2219.77)	1712.43 (2802.44)	**3.9881e-10** **(4.35113e-10)**	**2.52261e-09** **(2.40497e-09)**
f_3	15.8846 (9.53778)	31.3115 (46.2811)	**24.5038** **(40.7141)**	**4.62151e-10** **(5.1526e-10)**	**6.72887e-10** **(5.23577e-10**
f_4	55.8172 (17.0791)	**0.139295** **(0.853063)**	**0.0397984** **(0.196951)**	0.23879 (1.24822)	**0.0198992** **(0.140708)**
f_5	0.476326 (2.91565)	2.29725 (6.23978)	**7.24576e-14** **(2.50212e-14)**	**2.01927** **(5.62937)**	**3.96305e-14** **(8.27882e-15)**
f_6	0.0134364 (0.0169437)	0.139295 (0.853063)	**0.0624817** **(0.0569936)**	**0.0535327** **(0.0535873)**	**0.0524945** **(0.0498488)**
f_7	0.639873 (0.137024)	1.21987 (0.192725)	1.22987 (0.208248)	1.15387 (0.183181)	**1.19387** **(0.219842)**
f_8	-41.2553 (50.7622)	**-153.179** **(5.3920)**	**-153.365** **(2.16092)**	-150.644 (1.26064)	-150.326 (16.6553)
f_9	5.15659 (2.34525)	**0.855093** **(0.420195)**	0.920654 (0.434841)	**0.797715** **(0.413331)**	0.860774 (0.424196)
f_{10}	0.0181309 (0.126894)	**5.41382e-27** **(2.00007e-26)**	2.0807e-26 (5.06554e-26)	**4.46798e-28** **(3.52975e-28)**	**5.86288e-28** **(3.38157e-28)**

These results help to understand the contribution of each component to the overall performance of the algorithm. The predator-prey algorithm improves substantially on the results of the simple PSO for f_1, f_4 and f_8-f_{10}, with results slightly worse for the remaining functions. Overall, the predator-effect has a positive contribution since, for some functions, the improvement is of several orders of magnitude while, when a performance decrease occurs, is less marked.

When compared with the predator-prey optimizer, the introduction of an opposition learning based scout substantially improves the optimization results

for f_4, f_6 and, specially, for f_5. For the remaining functions the variation is mostly negligible, with exception of f_{10}. This illustrates the use of a scout particle that implements a general exploration behavior that is mostly beneficial for a specific problem. It also underlines that its inclusion does not substantially impact on the overall performance of the algorithm.

The local search scout, again in comparison with the PPO, has a more uniform effect. There is improvement in almost all functions, with particular focus on the unimodal ones. When there is no improvement the difference is negligible, strengthening the idea that the addition of scout particles doesn't disturb the general algorithm behavior. The local search scout performs a different role since, while the scouting is also general, i.e. not tailored for this particular problem, it seems to improve the algorithm performance over many problems.

Table 4. Experimental results for the problem specific scout particle: average and standard deviations of the best values found over 50 runs of an algorithm for each function; when the average is 0, the following value is the average number of function evaluations needed to find the optimum

	HMRPSO	FSDE	SPPO	SPPO+
f_1	7.3948e-26 (2.60705e-25)	1.04446 (7.38415)	0 [9200]	**0 [7250]**
f_2	3372.4 (10336.1)	13183.7 (9154.16)	6.95768e-29 (3.01847e-28)	**3.75725e-29** **(2.04331e-28)**
f_3	31.8958 (225.537)	158.062 (102.318)	1.07958e-28 (5.73612e-28)	**4.05921e-29** **(1.33317e-28)**
f_4	**0 [117000]**	40.1661 (37.248)	0.0198992 (0.140708)	0 [151600]
f_5	2.25064e-14 (2.2529e-14)	15.0721 (4.13056)	0 [16900]	**0 [15150]**
f_6	0.0322881 (0.0425845)	0.648994 (1.87741)	**0.0450732** **(0.062597)**	0.0471287 (0.0583248)
f_7	0.405873 (0.130008)	1.51993 (2.52228)	0.335876 (0.158765)	**0.293873** **(0.142012)**
f_8	-39.5094 (11.0244)	751.95 (395.148)	**-55.8945** **(10.2041)**	-53.3964 (9.4122)
f_9	2.70569 (1.11255)	7.32397 (2.4941)	0.0223451 (0.0842491)	**0.000777273** **(0.0038465)**
f_{10}	5.19578e-06 (1.98551e-05)	173886 (239596)	3.77984e-27 (4.94146e-27)	**3.87103e-28** **(1.38232e-27)**

For the third, and final, set of experiments, we changed the problems' characteristics, by moving the optimum to 0 in 80% of dimensions. For the remaining dimensions, 80% were set to x_{max} and the rest to a random value in the function's domain, which was also changed to $[0, x_{max}]$. A new problem is randomly generated for each run of an algorithm. These changes, in one hand, make the problems harder for evolutionary algorithms, by placing the solution in the border of the search space for most of the dimensions. This allows to better evaluate the robustness of the algorithms being compared. On the other hand, this knowledge about the optimum's characteristics can be used to test the use of a scout

built around specific problem knowledge. In these experiments we used a simple scout that, for each iteration, moved the particle, in a random dimension, to 0 (with 80% probability) or x_{max} (with 20% probability). The results for these experiments are presented in Table 4, where SPPO+ represents the scouting predator-prey algorithm with the extra scout. In this table, when the algorithm found the global optimum in all runs, we present in brackets the average number of used function evaluations, instead of the standard deviation.

Looking at the results in Table 4, our first conclusion is that the problems' new formulation was particularly challenging for the FSDE algorithm, which struggled considerably to optimize many of the benchmark functions. The HMRPSO algorithm performance decayed more gracefully, inclusively achieving the best results for f_4, but it also had great difficulties to optimize some functions, specially f_2, f_3, f_9 and f_{10}. The SPPO algorithm performed at least as well as for the previous formulations, which suggests this algorithm to be more robust than both the FSDE and HMRPSO. The results for SPPO+, using the third scout particle, were superior in 8 of the 10 benchmark functions, when compared with those of the SPPO. These results illustrate the fact that even a single scout particle, using a specific behavior based on problem specific knowledge, can have a positive impact on the algorithm's performance. Additionally, using scout particles, this impact can be achieved without performing global changes to the algorithm or using a large number of extra function evaluations.

4 Conclusions

In this paper we discussed the use of scout particles to improve the performance of an heterogeneous particle swarm optimizer algorithm. We added three different scout particles to the swarm, performing different roles, and thus contributing in different ways to the final behavior of the swarm. We used a local search based scout to improve the overall optimization performance of the algorithm and observed that an opposition learning based scout has a more limited effect, which still could have a large impact for specific functions. Finally we changed the experimental conditions and built a scout based on the knowledge about that changes to adjust the algorithm to the new conditions, with additional gains in performance.

Overall, we can conclude that scout particles can be easily used to add new exploratory behaviors to the algorithm, without perturbing the general dynamic of the swarm, either by introducing global modifications or using extra resources, e.g. objective function evaluations. From the extensive experiments performed on the benchmark functions, it is apparent that the resulting scouting predator-pray optimizer is not only competitive but even superior to other state-of-the-art evolutionary, not only in optimization performance but also in robustness.

References

1. Angeline, P.J.: Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 601–610. Springer, Heidelberg (1998)

2. Beekman, M., Gilchrist, A., Duncan, M., Sumpter, D.: What makes a honeybee scout? Behavioral Ecology and Sociobiology 61, 985–995 (2007)
3. Beyer, H.-G., Schwefel, H.-P.: Evolution strategies - a comprehensive introduction. Natural Computing 1, 3–52 (2002)
4. Engelbrecht, A.P.: Heterogeneous Particle Swarm Optimization. In: Dorigo, M., Birattari, M., Di Caro, G.A., Doursat, R., Engelbrecht, A.P., Floreano, D., Gambardella, L.M., Groß, R., Şahin, E., Sayama, H., Stützle, T. (eds.) ANTS 2010. LNCS, vol. 6234, pp. 191–202. Springer, Heidelberg (2010)
5. Gao, H., Xu, W.: A new particle swarm algorithm and its globally convergent modifications. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41(5), 1334–1351 (2011)
6. Gao, H., Xu, W.: Particle swarm algorithm with hybrid mutation strategy. Applied Soft Computing 11(8), 5129–5142 (2011)
7. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
8. Montes de Oca, M., Pena, J., Stutzle, T., Pinciroli, C., Dorigo, M.: Heterogeneous particle swarm optimizers. In: IEEE Congress on Evolutionary Computation, CEC 2009, pp. 698–705 (May 2009)
9. Omran, M.G.H., Engelbrecht, A.P.: Free search differential evolution. In: Proc. of the 11th Congress on Evolutionary Computation, CEC 2009, pp. 110–117. IEEE Press, Piscataway (2009)
10. Petalas, Y., Parsopoulos, K., Vrahatis, M.: Memetic particle swarm optimization. Annals of Operations Research 156, 99–127 (2007)
11. Poli, R.: Analysis of the publications on the applications of particle swarm optimisation. J. Artif. Evol. App., 4:1–4:10 (January 2008)
12. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence 1, 33–57 (2007)
13. Rahnamayan, S., Tizhoosh, H., Salama, M.: Opposition-based differential evolution. IEEE Transactions on Evolutionary Computation 12(1), 64–79 (2008)
14. Shi, Y., Eberhart, R.: Empirical study of particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999, vol. 3, pp. (xxxvii+2348) (1999)
15. Silva, A., Neves, A., Costa, E.: An Empirical Comparison of Particle Swarm and Predator Prey Optimisation. In: O'Neill, M., Sutcliffe, R.F.E., Ryan, C., Eaton, M., Griffith, N.J.L. (eds.) AICS 2002. LNCS (LNAI), vol. 2464, pp. 103–110. Springer, Heidelberg (2002)
16. Silva, A., Neves, A., Gonçalves, T.: An Heterogeneous Particle Swarm Optimizer with Predator and Scout Particles. In: Kamel, M., Karray, F., Hagras, H. (eds.) AIS 2012. LNCS, vol. 7326, pp. 200–208. Springer, Heidelberg (2012)
17. Storn, R., Price, K.: Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359 (1997)
18. Tizhoosh, H.R.: Opposition-based learning: A new scheme for machine intelligence. In: Proc. of the Int. Conf. on Computational Intelligence for Modelling, Control and Automation, CIMCA 2005, Washington, DC, pp. 695–701 (2005)
19. Wang, H., Li, H., Liu, Y., Li, C., Zeng, S.: Opposition-based particle swarm algorithm with cauchy mutation. In: IEEE Congress on Evolutionary Computation, CEC 2007, pp. 4750–4756 (September 2007)

QR-DCA: A New Rough Data Pre-processing Approach for the Dendritic Cell Algorithm

Zeineb Chelly and Zied Elouedi

LARODEC, University of Tunis, 2000 Le Bardo, Tunisia
zeinebchelly@yahoo.fr, zied.elouedi@gmx.fr

Abstract. In this paper, we propose a new approach of data pre-processing based on rough set theory for the Dendritic Cell Algorithm (DCA). Our hybrid immune inspired model, denoted QR-DCA, is based on the functioning of dendritic cells within the framework of rough set theory and more precisely, on the QuickReduct algorithm. As the DCA data pre-processing phase is divided into two sub-steps, feature selection and signal categorization, our QR-DCA model selects the right features for the DCA classification task and categorizes each one of them to its specific signal category. This is achieved while preserving the same DCA main characteristic which is its lightweight in terms of running time. Results show that our new approach generates good classification results. We will also compare our QR-DCA to other rough DCA models to show that our new approach outperforms them in terms of classification accuracy while keeping the worthy characteristics expressed by the DCA.

Keywords: Artificial immune systems, Dendritic cells, Rough sets, QuickReduct.

1 Introduction

Artificial Immune Systems (AIS) are a class of computationally intelligent systems inspired by the principles of the natural immune system. As AIS is being developed significantly, novel algorithms termed "2nd Generation AISs" have been created. One such 2nd Generation AIS is the Dendritic Cell Algorithm (DCA) [1] which is derived from behavioral models of natural dendritic cells (DCs). As a binary classifier, the DCA performance is mainly based on its data pre-processing phase which is divided into two main phases which are feature selection and signal categorization. More precisely and for data pre-processing, DCA uses the Principal Component Analysis (PCA) to automatically select features and to categorize them to their specific signal types; as danger signals (DS), as safe signals (SS) or as pathogen-associated molecular patterns (PAMP) [2]. DCA combines these signals with location markers in the form of antigen to perform antigen classification. For signal selection, PCA transforms a finite number of possibly correlated vectors into a smaller number of uncorrelated vectors, termed "principal components" which reveal the internal structure of the given data with the focus on data variance [2]. Nevertheless, using PCA as a dimensionality reduction technique presents some shortcomings as it is not necessarily true

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 140–150, 2013.

that the first selected principal components that capture most of the variance are the adequate features to retain [3]. Consequently, choosing these components for the DCA may influence its classification task by producing unreliable results. For feature categorization, DCA uses the PCA ranking of attributes which is based on variability and maps this obtained order to the ranking of the signal categories of the DCA which is in the order: Safe, PAMP and Danger; implying the significance of each signal category to the signal transformation of the DCA [2]. However, this categorization reasoning which is based on attributes' ranking and where the variability of attributes is equivalent to importance could not be considered as a coherent and consistent categorization procedure. Hence, in [4], a first work, named RC-DCA, was developed to solve these issues. RC-DCA is based on Rough Set Theory (RST) to perform data pre-processing. It is based on the reduct and the core RST fundamental concepts to select the most important features and to categorize them to their specific signal types. In [4], it was shown that applying RST, instead of PCA, is more appropriate for the DCA data pre-processing phase leading to a better binary classifier. However, to select the right set of features, RC-DCA generates all possible subsets and retrieves those with a maximum rough set dependency degree. Obviously, this is an expensive solution to the problem and is only practical for very simple data sets. Most of the time only one reduct is required as, typically, only one subset of features is used to reduce a data set, so all the calculations involved in discovering the rest are pointless. In addition, this time consuming task led to neglect the main DCA characteristic which is its lightweight in terms of running time [5]. Thus, in this paper, we propose a novel bio-inspired hybrid model of the DCA based on a new signal selection and categorization technique. Our new model, named QR-DCA, is based on the behavior of natural dendritic cells and grounded on the framework of rough set theory for data pre-processing where it adopts the QuickReduct algorithm. The main contributions of this paper are to introduce the concept of RST, specifically the QuickReduct algorithm, in the DCA data pre-processing phase and to show how our proposed new model, QR-DCA, can find a trade-off between generating good classification results and preserving the lightweight of the algorithm. We, also, aim to compare the results obtained from QR-DCA to other rough DCA models proposed in literature.

2 The Dendritic Cell Algorithm

1)Introducing Dendritic Cells: DCs are antigen presenting cells that possess the ability to capture and process antigens [6]. DCs differentiate into three main states upon the receipt of signals: PAMPs, danger, safe. The first DC maturation state is the immature state (iDCs). The differentiation of iDCs depends on the combination of the signals received leading either to a full maturation state or to a partial maturation state. Under the reception of safe signals, iDCs migrate to the semi-mature state (smDCs) causing antigens tolerance. iDCs migrate to the mature state (mDCs) if they are more exposed to danger signals and PAMPs than safe signals. mDCs present the collected antigens in a dangerous context.

2)Abstract View of the Dendritic Cell Algorithm: The initial step of the DCA is data pre-processing where feature selection and signal categorization are achieved. More precisely, DCA selects the most important features, from the initial input database, and assigns each selected attribute to its specific signal category (SS, DS or PAMP). To do so, DCA applies the PCA [7].

For signal selection, PCA selects the first "principal components" which reveal the internal structure of the given data with the focus on data variance. PCA reduces data dimension, by accumulating the vectors that can be linearly represented by each other [2]. Once features are selected, PCA is applied to assign each attribute to its specific signal type. In fact, PCA performs a ranking procedure by using a sum of the absolute values of the weights used for signal transformation by the DCA. Once ranking is performed, the attributes are mapped into the DCA input signal categories, by correlating the PCA ranking with the ranking of signal categories - which implies the significance of each signal type to the signal transformation of the DCA - which is in the order: Safe, PAMP, and Danger [2]. DCA adheres these signals and antigen to fix the context of each object (DC) which is the step of Signal Processing. The algorithm processes its input signals in order to get three output signals: costimulation signal (Csm), semi-mature signal ($Semi$) and mature signal (Mat). A migration threshold is incorporated into the DCA in order to determine the lifespan of a DC. As soon as the Csm exceeds the migration threshold; the DC ceases to sample signals and antigens. The migration state of a DC to the semi-mature state or to the mature state is determined by the comparison between cumulative $Semi$ and cumulative Mat. If the cumulative $Semi$ is greater than the cumulative Mat, then the DC goes to the semi-mature context, which implies that the antigen data was collected under normal conditions. Otherwise, the DC goes to the mature context, signifying a potentially anomalous data item. This step is known to be the Context Assessment phase. The nature of the response is determined by measuring the number of DCs that are fully mature and is represented by the Mature Context Antigen Value (MCAV). $MCAV$ is applied in the DCA final step which is the Classification step and used to assess the degree of anomaly of a given antigen. The closer the $MCAV$ is to 1, the greater the probability that the antigen is anomalous. Those antigens whose $MCAV$ are greater than the anomalous threshold are classified as anomalous while the others are classified as normal. More DCA details and its pseudocode can be found in [1].

3 Rough Set Theory

In RST [8], an *information table* is defined as a tuple $I = (U, A)$ where U is non-empty set of primitive objects and A is non-empty set of attributes. A may be partitioned into C and D, called *condition* and *decision* attributes, respectively. With any $P \subset A$ there is an associated equivalence relation $IND(P)$ defined as: $IND(P) = \{(x, y) \in U^2 : \forall a \in P, a(x) = a(y)\}$, where $a(x)$ denotes the value of feature a of object x. The family of all equivalence classes of $IND(P)$

is denoted by $U/IND(P)$. The equivalence classes of the P-indiscernibility relation are denoted $[x]_P$. Let $X \subseteq U$, the P-lower approximation of a set can now be defined as: $\underline{P}(X) = \{x|[x]_P \subseteq X\}$. The P-lower approximation is the set of objects U that are surely in X. Let P and Q be equivalence relations over U, then the positive region can be defined as: $POS_P(Q) = \bigcup_{X \in U/Q} \underline{P}(X)$. The positive region contains all objects of U that can be classified to classes of U/Q using the knowledge in attributes P. An important issue in data analysis is discovering dependencies between attributes. Intuitively, a set of attributes Q depends totally on a set of attributes P, denoted $P \Rightarrow Q$, if all attribute values from Q can be uniquely determined by values of attributes from P. In particular, if there exists a functional dependency between values of Q and P, then Q depends totally on P. Dependency can be defined in the following way: For P, $Q \subset A$, Q depends on P in a degree k $(0 \leq k \leq 1)$, denoted $P \Rightarrow_k Q$, if $k = \gamma_P(Q) = |POS_P(Q)|/|U|$; If k = 1 Q depends totally on P, if $k < 1$ Q depends partially (in a degree k) on P, and if k = 0 Q does not depend on P. RST performs the reduction of attributes by comparing equivalence relations generated by sets of attributes. Attributes are removed so that the reduced set provides the same quality of classification as the original. A *reduct* is defined as a subset R of the conditional attribute set C such that $\gamma_R(D) = \gamma_C(D)$. Note that a given data set may have many attribute reduct sets. The intersection of all the sets in R is called the *core*, reflecting those attributes that cannot be eliminated without introducing more contradictions to the data set. In RST, a *reduct* with minimum cardinality is searched for; in other words an attempt is made to locate a single element of the minimal reduct set. A basic way of achieving this is to calculate the dependencies of all possible subsets of C. Any subset X with $\gamma_X(D) = 1$ is a reduct; the smallest subset with this property is a minimal reduct. However, for large data sets this method is impractical and an alternative strategy is required. One possible way to avoid these extra calculations, is to apply the QuickReduct Algorithm [9] that attempts to calculate a minimal reduct without exhaustively generating all possible subsets. It starts off with an empty set and adds in turn those attributes that result in the greatest increase in $\gamma_P(Q)$, until this produces its maximum possible value for the data set (usually 1). An illustrative example of the QuickReduct Algorithm application as well as its pseudocode can be found in [9].

4 QR-DCA: The Solution Approach

In this Section, we present our QR-DCA model based on RST, and specifically on the QuickReduct algorithm, for the automatic DCA data pre-processing phase including feature selection and signal categorization.

4.1 The QR-DCA Signal Selection Process

For antigen classification, our learning problem has to select high discriminating features from the original input database which corresponds to the antigen

information data set. We may formalize this problem as an information table, where universe $U = \{x_1, x_2, \ldots, x_N\}$ is a set of antigen identifiers, the conditional attribute set $C = \{c_1, c_2, \ldots, c_A\}$ contains each feature of the information table to select and the decision attribute D of our learning problem corresponds to the class label of each sample.

For feature selection, QR-DCA computes, first of all, the dependency of the entire database $\gamma_C(D)$. To do so, QR-DCA has to calculate the positive region for the whole attribute set C: $POS_C(D)$ (as presented in Section 3). Once the consistency of the database is measured, QR-DCA starts off with an empty set and moves to calculate the dependency of each attribute c a part: $\gamma_c(D)$. The attribute c having the greatest value of dependency is added to the empty set. Once the first attribute c is selected, QR-DCA adds, in turn, one attribute to the selected first attribute and computes the dependency of each obtained attributes' couple $\gamma_{\{c,c_i\}}(D)$. The algorithm chooses the couple having the greatest dependency degree. The process of adding each time one attribute to the subset of the selected features continues until the dependency of the obtained subset equals the consistency of the entire database already calculated: $\gamma_C(D)$.

The generated subset of the selected features, constituting the reduct, shows the way of reducing the dimensionality of the original data set by eliminating those conditional attributes that do not appear in the set. Those discarded attributes are removed in each QR-DCA computation level since they do not add anything new to the target concept nor help the QR-DCA to perform well its classification task. However, the reduct features represent the most informative features that preserve nearly the same classification power of the original data set. Using the reduct concept, our method can guarantee that attributes of extracted feature patterns will be the most relevant for its classification task.

4.2 The QR-DCA Signal Categorization Process

The second step of our QR-DCA model data pre-processing phase is feature categorization. More precisely, our method has to assign for each selected attribute, produced by the previous step and which is included in the generated reduct, its definite and specific signal category. The general guidelines for signal categorization are as follows:

- **Safe signals :** Their presence certainly indicates that no anomalies are present.
- **PAMPs :** Their presence usually means that there is an anomalous situation.
- **Danger signals :** Their presence may or may not show an anomalous situation, however the probability of an anomaly is higher than under normal circumstances.

From the previous definitions, both PAMP and SS are positive indicators of an anomalous and normal signal while the DS is measuring situations where the risk of anomalousness is high, but there is no signature of a specific cause. In other words, PAMP and SS have a certain final context (either an anomalous or a normal behavior) while the DS cannot specify exactly the final context to assign to the collected antigen. This is because the information returned by the

DS is not certain as the collected antigen may or may not indicate an anomalous situation. This problem can be formulated as follows:

Both PAMP and SS are more informative than DS which means that both of these signals can be seen as indispensable attributes. To represent this level of importance, our method uses the first obtained couple of features through the reduct generation. On the other hand, DS is less informative than PAMP and SS. Therefore, our method applies the rest of the reduct attributes, discarding the two first selected attributes that are chosen to represent the SS and PAMP signals, to represent the DS. More precisely, our method processes as follows:

As QR-DCA has already calculated the dependency of each attribute c a part, $\gamma_c(D)$, QR-DCA selects the attribute c having the greatest dependency degree to form the SS as it is considered the most informative feature added to the reduct. With no additional computations and since QR-DCA has already computed the dependency of each attributes' couple $\gamma_{\{c,c_i\}}(D)$ when adding, in turn, one attribute c_i to the selected first attribute c that represents the SS, QR-DCA chooses the couple having the greatest dependency degree. More precisely, QR-DCA selects that second attribute c_i to form the PAMP signal. And finally, the rest of the reduct attributes are combined and affected to represent the DS as it is less than certain to be anomalous.

Once the selected features are assigned to their suitable signal types, our method calculates the values of each signal category using the same process as the standard DCA [1]. The output is thus a new information table which reflects the signal database. In fact, the universe U of the induced signal data set is $U = \{x_1^{'}, x_2^{'}, \ldots, x_N^{'}\}$ a set of antigen identifiers and the conditional attribute set $C = \{SS, PAMP, DS\}$ contains the three signal types: SS, PAMPs and DS. Once data pre-processing is achieved, QR-DCA processes its next steps which are the Signal Processing, the Context Assessment and the Classification step as the DCA does and as described in Section 2.

5 Experimental Setup

To test the validity of our QR-DCA hybrid model, our experiments are performed on two-class databases from [10] described in Table 1. We try to show that our QR-DCA can find a trade-off between generating good classification results and having a lightweight in terms of running time. Thus, we will compare the QR-DCA performance to our first work, RC-DCA, published in [4] which is also based on RST for the DCA data pre-processing phase. The common idea between QR-DCA and RC-DCA is to assign for each selected feature a specific signal category: either as SS, DS or PAMP. Nevertheless, RC-DCA generates all the possible reducts, which is a time consuming task, to select the reduct having the minimal set of features among the other generated reducts. In addition, RC-DCA differs from our new rough DCA model in the categorization step which focuses on proposing different solutions in case where this method produces one reduct or a family of reducts. More details about RC-DCA can be found in [4]. The QR-DCA performance is also compared to another rough DCA work,

named RST-DCA [11]. The main difference between RST-DCA and both QR-DCA and RC-DCA is that RST-DCA assigns only one attribute to form both SS and PAMP as they are seen as the most important signals. As for the DS categorization, RST-DCA combines the rest of the reduct features and assigns the resulting value to the DS. Like RC-DCA, RST-DCA generates all the possible reducts and proposes solutions to handle both cases (generating one reduct or a family of reducts) for data pre-processing. More details about RST-DCA can be found in [11].

For data pre-processing and for all the mentioned rough DCA works including QR-DCA, each data item is mapped as an antigen, with the value of the antigen equal to the data ID of the item. In all experiments, a population of 100 cells is used and 10 DCs sample the antigen vector each cycle. To perform anomaly detection, a threshold which is automatically generated from the data is applied to the MCAVs. The MCAV threshold is derived from the proportion of anomalous data instances of the whole data set. Items below the threshold are classified as class 1 and above as class 2. The resulting classified antigens are compared to the labels given in the original data sets. For each experiment, the results presented are based on mean MCAV values generated across 10 runs.

We evaluate the performance of the rough DCA methods in terms of number of extracted features, running time, and accuracy which is defined as: $Accuracy = (TP + TN)/(TP + TN + FN + FP)$; where TP, FP, TN, and FN refer respectively to: true positive, false positive, true negative and false negative. We will also compare the classification performance of our QR-DCA method to well known classifiers which are the Support Vector Machine (SVM), Artificial Neural Network (ANN) and the Decision Tree (DT).

Table 1. Description of Databases

Database	Ref	♯ Instances	♯ Attributes
Spambase	SP	4601	58
SPECTF Heart	SPECTF	267	45
Cylinder Bands	CylB	540	40
Chess	Ch	3196	37
Ionosphere	IONO	351	35
Congressional Voting Records	CVT	435	17
Tic-Tac-Toe Endgame	TicTac	958	10

6 Results and Analysis

Let us remind that the first step of the standard DCA classification algorithm is data pre-processing which is based on the use of PCA. In [11] and [4] and by the development of both RST-DCA and RC-DCA, we have proved that applying PCA for both feature selection and signal categorization is not convenient for the DCA as both phases are not consistent. We have also shown that applying rough set theory with DCA is a good alternative leading to a better classification

performance. However, these two rough DCA models suffer from some limitations; principally the time taken by the algorithms to process which contradicts the main characteristic of the standard DCA: its lightweight in terms of running time. Thus, in this paper, we have developed a new rough DCA hybrid model, QR-DCA, where we show that this proposed solution can find the trade-off between generating good classification results and processing in less time than both RC-DCA and RST-DCA. This is confirmed by the results presented in Table 2. From these results, we will also show that assigning for each selected feature a specific signal category, a process performed by both QR-DCA and RC-DCA, lead to a better performance than assigning the same attribute to both SS and PAMP, a process performed by RST-DCA.

Table 2. Comparison Results of the Rough DCA Approaches

Database	Accuracy (%)			♯ Attributes			Time (s)		
	DCA			DCA			DCA		
	QR	RC	RST	QR	RC	RST	QR	RC	RST
SP	98.87	98.45	94.5	11	8	8	1976.05	3184.83	2923.41
SPECTF	93.26	88.38	82.4	12	4	4	5.49	1423.02	1361.77
CylB	97.46	97.46	96.67	7	7	7	12.68	1441.93	1398.12
Ch	98.84	98.84	98.02	11	11	11	571.05	1779.83	1697.01
IONO	96.58	97.15	96.29	22	19	19	15.88	668.32	591.13
CVT	97.93	98.85	96.55	11	8	8	7.03	17.83	10.54
TicTac	96.65	95.3	93.52	8	6	6	49.89	62.66	58.80

From Table 2, we can notice that RST-DCA and RC-DCA models have the same number of selected features. This is explained by the fact that both models are based on the same feature selection phase. They generate all the possible reducts and choose the one having the smaller number of features. However, our QR-DCA new version has either the same number of features as both RST-DCA and RC-DCA or more features. This is explained by the fact that QR-DCA, by applying the QuickReduct algorithm, follows the features' path that generates the highest dependency degree. Consequently, the taken path may either lead to a final reduct including the smallest number of features or to a path including more selected features; but still this obtained reduct includes the most important features to retain. For instance, applying QR-DCA to the IONO database, the number of selected attributes is 22. However, when applying RST-DCA or RC-DCA, the number of selected features is 19. Applying the three rough DCA models to the Ch database, the number of the selected features is the same: 11. We have also to mention that obtaining the same number of features does not mean that this reduct includes the same attributes; the attributes may differ.

Based on these selected attributes, the accuracies of the algorithms are calculated. From Table 2, we can notice that the difference between the classification accuracies generated by both QR-DCA and RC-DCA is not significant. Thus,

we can say that QR-DCA has nearly the same classification performance as RC-DCA. For instance, when applying the algorithms to the SP data set, the classification accuracy of QR-DCA is set to 98.87% and when applying RC-DCA to the same database, the accuracy is set to 98.45%. In some databases, the QR-DCA classification accuracy is a bit less than the one generated by RC-DCA and sometimes a bit higher. We also remark, that in all databases, QR-DCA and RC-DCA outperform the classification accuracy generated by RST-DCA. For instance, the classification accuracy of RST-DCA when applied to the SP database is set to 94.5% which is less than 98.87% and 98.45% generated respectively by QR-DCA and RC-DCA. This is explained by the fact that RST-DCA differs from QR-DCA and RC-DCA in the signal categorization phase. Both QR-DCA and RC-DCA assign different features to different signal categories (DS, SS, PAMP). However, RST-DCA uses the same attribute to assign it for both SS and PAMP. As for the DS categorization, RST-DCA combines the rest of the reduct features to assign it for the DS. From these results, we can conclude that it is crucial to assign for each signal category a specific and different feature.

Another advantage of our QR-DCA is that it takes less time to process than RC-DCA and RST-DCA. This is confirmed by the results appearing in Table 2. For example, when applying the algorithms to the CylB database, the amount of time taken by QR-DCA to process is 12.68(s) which is less than the times taken by RC-DCA and RST-DCA which are 1441.93(s) and 1398.12(s), respectively. The QR-DCA lightweight in terms of running time is explained by the advantage of using the QuickReduct algorithm as it attempts to calculate a reduct without exhaustively generating all possible subsets. In contrast, both RST-DCA and RC-DCA generate all possible subsets and retrieve those with a maximum rough set dependency degree. Obviously, this is an expensive solution to the problem. Most of the time only one reduct is required as, typically, only one subset of features is used to reduce a data set, so all the calculations involved in discovering the rest are pointless.

We have also compared the performance of our QR-DCA to other classifiers including SVM, ANN and DT. The comparison made is in terms of the average of accuracies on the databases presented in Table 1. The parameters of SVM, ANN and DT are set to the most adequate parameters to these algorithms using the Weka software. Figure 1 shows that our QR-DCA has nearly the same classification performance as RC-DCA. It also shows that QR-DCA outperforms RST-DCA, SVM, ANN and DT.

To summarize, QR-DCA is a good classification technique proposed as an alternative to our RC-DCA first work. QR-DCA has the advantage of generating good classification results while preserving a lightweight in terms of running time. We have also shown that it is crucial that DCA assigns different attributes for each signal type. QR-DCA performs much better than the mentioned classifiers in terms of classification accuracy.

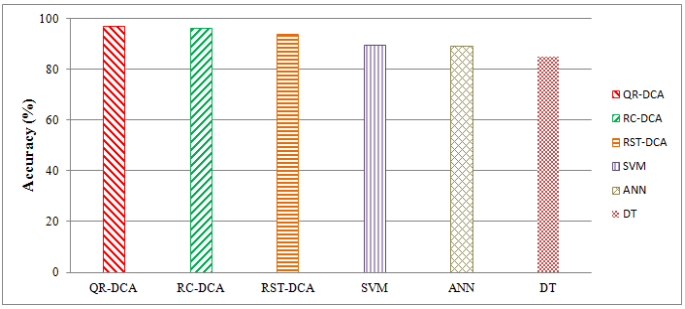


Fig. 1. Comparison of Classifiers' Average Accuracies on the 7 Binary Data sets

7 Conclusion and Further Work

In this paper, we have introduced a new hybrid DCA bio-inspired model based on RST. Our model aims to select the convenient set of features and to perform their signal categorization using the QuickReduct algorithm. Results show that our method is capable of finding a trade-off between generating good classification results and keeping the algorithm lightweight in terms of running time. As future work we aim to apply the fuzzy RST for the DCA data pre-processing phase.

References

1. Greensmith, J., Aickelin, U., Cayzer, S.: Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection. In: Jacob, C., Pilat, M.L., Bentley, P.J., Timmis, J.I. (eds.) ICARIS 2005. LNCS, vol. 3627, pp. 153–167. Springer, Heidelberg (2005)
2. Gu, F., Greensmith, J., Oates, R., Aickelin, U.: Pca 4 dca: The application of principal component analysis to the dendritic cell algorithm. CoRR (2010)
3. Cantú-Paz, E.: Feature Subset Selection, Class Separability, and Genetic Algorithms. In: Deb, K., Tari, Z. (eds.) GECCO 2004, Part I. LNCS, vol. 3102, pp. 959–970. Springer, Heidelberg (2004)
4. Chelly, Z., Elouedi, Z.: RC-DCA: A New Feature Selection and Signal Categorization Technique for the Dendritic Cell Algorithm Based on Rough Set Theory. In: Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M. (eds.) ICARIS 2012. LNCS, vol. 7597, pp. 152–165. Springer, Heidelberg (2012)
5. Greensmith, J., Aickelin, U.: The Deterministic Dendritic Cell Algorithm. In: Bentley, P.J., Lee, D., Jung, S. (eds.) ICARIS 2008. LNCS, vol. 5132, pp. 291–302. Springer, Heidelberg (2008)
6. Lotze, M.T., Thomson, A.W.: Dendritic Cells: Biology and Clinical Applications, 2nd edn., vol. 794 (2001)
7. Jolliffe, I.T.: Principal component analysis. Springer, New York (2002)

8. Pawlak, Z.: Rough sets. International Journal of Computer and Information Science 11, 341–356 (1982)
9. Jensen, R., Shen, Q.: A rough set-aided system for sorting www bookmarks. In: Zhong, N., et al. (eds.) Web Intelligence: Research and Development, pp. 95–105 (2001)
10. Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)
11. Chelly, Z., Elouedi, Z.: RST-DCA: A Dendritic Cell Algorithm Based on Rough Set Theory. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part III. LNCS, vol. 7665, pp. 480–487. Springer, Heidelberg (2012)

Convergence Rates of Evolutionary Algorithms for Quadratic Convex Functions with Rank-Deficient Hessian

Günter Rudolph

Technische Universität Dortmund, Fakultät für Informatik
Otto-Hahn-Str. 14, 44227 Dortmund, Germany

Abstract. The best achievable convergence rates of mutation-based evolutionary algorithms are known for various characteristic test problems. Most results are available for convex quadratic functions with Hessians of full rank. Here, we prove that linear convergence rates are achievable for convex quadratic functions even though the Hessians are rank-deficient. This result has immediate implications for recent convergence results for certain evolutionary algorithms for bi-objective optimization problems.

Keywords: evolutionary algorithm, convergence rate, quadratic functions, rank-deficient Hessian.

1 Introduction

The maximum convergence speed of mutation-based evolutionary algorithms (EA) have been derived for various characteristic test problems $f : \mathbb{R}^n \to \mathbb{R}$. In 1973 Rechenberg [15] began this kind of research by deriving the convergence rate the $(1 + 1)$-EA for the so-called sphere and corridor model in case of Gaussian mutations. This work was extended by others [21, 20, 3, 6, 7, 2, 16–19, 14, 13, 8] in various directions including the type of evolutionary algorithms, test problems and mutation distributions. These theoretical analyses typically assumed that the step size or mutation distribution can be adjusted optimally by exploiting knowledge about the distance to the optimal solution or optimal value. Jägersküpper's results [11, 12] extended the theory considerably by proving that the step sizes or variances of the mutations controlled by mechanisms in the spirit of Rechenberg's $\frac{1}{5}$-success rule [15] also lead to linear convergence rates for quadratic convex functions with full rank Hessians. Objective functions of this type exhibit positive definite Hessian matrices. Here, we shall consider the case of rank-deficient Hessians which implies that the Hessian is positive semidefinite and not positive definite. This scenario is analyzed in [5] with the result of an expected convergence rate $\in [c, 1]$ for some $c \in (0, 1)$. The inclusion of $c = 1$ restricts the power of the result considerably since we have no progress in convergence then. The limitation in that result stems from the necessity of a positive least eigenvalue of the Hessian in the proof technique. Since the least eigenvalue of a positive semidefinite and rank-deficient matrix is

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 151–160, 2013.

zero another proof technique is required. The application of this alternative technique leads to the result that the case $c = 1$ is actually excluded. This result is a further step towards a comprehensive convergence theory of EAs.

In section 2 we introduce basic results from linear algebra, the problem class and the evolutionary algorithm under consideration. Section 3 contains the convergence analysis whereas section 4 discusses implications of the result for the multiobjective setting. Conclusions are presented in section 5.

2 Preliminaries

2.1 Tools from Linear Algebra

The terminology and results are extracted from [1].

Definition 1. *A matrix $A : n \times n$ is termed*
a) symmetric *if $A = A'$, where A' denotes the transpose of A,*
b) orthogonal *if $A'A = I_n$, where I_n is the unit matrix of dimension n,*
c) positive definite *(p.d.) if $x'Ax > 0$ for all $x \in \mathbb{R}^n \setminus \{0\}$,*
d) positive semidefinite *(p.s.d.) if $x'Ax \geq 0$ for all $x \in \mathbb{R}^n$,*
e) rank-deficient *if* $\mathsf{rank}\ A < n$. □

The result below will serve as a central tool to simplify the analysis in section 3.

Theorem 1. *Let $A : n \times n$ be a p.s.d. symmetric matrix with* $\mathsf{rank}A < n$.
a) A is diagonizable.
b) All eigenvalues are nonnegative.
c) The number of positive eigenvalues is equal to the rank of A.
d) There exists an orthogonal matrix T such that

$$T'AT \;=\; \mathsf{diag}(\nu_1, \ldots, \nu_r, 0, \ldots, 0)$$

where $\nu_1 \leq \nu_2 \leq \ldots, \leq \nu_r$ are the positive eigenvalues and $r = \mathsf{rank}\ A$. □

2.2 Problem Class

In section 3 we shall analyze the problem class $\min\{f(x) : x \in \mathbb{R}^n\}$ with $f(x) = x'Ax$ where $A : n \times n$ is symmetric, positive semidefinite and rank-deficient with $\mathsf{rank}\ A = r < n$. Since the Hessian matrix $\nabla^2 f(x) = 2\,A$ is p.s.d. for all $x \in \mathbb{R}^n$ by precondition, all $x \in \mathbb{R}^n$ with $\nabla f(x) = 2\,Ax = 0$ are global minimizers. Thus, optimal solutions x^* are elements of the null space or kernel of matrix A, i.e., $x^* \in \mathsf{kern}\ A = \{x \in \mathbb{R}^n : Ax = 0\}$. Since $\mathsf{dim}(\mathsf{kern}\ A) + \mathsf{rank}\ A = n$ in general and $\mathsf{rank}\ A = r < n$ by assumption, the dimension of the null space is $n - r \geq 1$. As a consequence, this problem has denumerable many solutions.

2.3 Algorithm

The analysis will be restricted to the $(1+1)$-EA sketched in Alg. 1. Here, $Z \sim N(0, I_n)$ means that Z is a random vector of dimension n with zero expectation vector and unit matrix as covariance matrix. It can be generated by drawing n independent, standard normally distributed random numbers $Z_1, \ldots, Z_n$ and setting $Z = (Z_1, Z_2, \ldots, Z_n)'$. The choice of σ_t will be a result of the analysis in section 3.

Algorithm 1. $(1+1)$-EA

```
choose X^(0) ∈ R^n at random; set t = 0
repeat
    draw Z ~ N(0, I_n)
    Y = X^(t) + σ_t · Z
    if f(Y) ≤ f(X^(t)) then
        X^(t+1) = Y
    else
        X^(t+1) = X^(t)
    end if
    t = t + 1
until stopping criterion fulfilled
```

2.4 Convergence Rate

To avoid misunderstandings it is necessary to equip the notion of "convergence" with a precise meaning.

Definition 2. *Let X be a random variable and $(X_k)_{k\in\mathbb{N}_0}$ a sequence of random variables defined on a probability space $(\Omega, \mathcal{A}, P)$. Then (X_k) is said to*

(a) converge completely to X, if for any $\epsilon > 0$

$$\lim_{k\to\infty} \sum_{j=1}^{k} P\{|X_j - X| > \epsilon\} < \infty;$$

(b) converge almost surely or with probability 1 to X, if

$$P\{\lim_{k\to\infty} |X_k - X| = 0\} = 1;$$

(c) converge in probability to X, if for any $\epsilon > 0$

$$\lim_{k\to\infty} P\{|X_k - X| > \epsilon\} = 0;$$

(d) converge in mean to X, if

$$\lim_{t\to\infty} E(|X_k - X|) = 0. \qquad \square$$

The velocity of approaching a limit is expressed by the "convergence rate."

Definition 3. *Let $(Z_k : k \geq 0)$ be a non-negative random sequence. The sequence is said to converge geometrically fast in mean (in probability, w.p. 1) to zero if there exists a constant $q > 1$ such that the sequence $(q^k Z_k : k \geq 0)$ converges in mean (in probability, w.p. 1) to zero. Let $q^* > 1$ be supremum of all constants $q > 1$ such that geometrically fast convergence is still guaranteed. Then $c = 1/q$ is called the* convergence rate. *A sequence with geometrically fast convergence is synonymously denoted to have a* linear convergence rate. □

Let $\rho(\cdot)$ denote a function that measures the performance of an EA's population X_k and ρ^* the target value. If the sequence $(Z_k)_{k\geq 0}$ defined by $Z_k = |\rho(X_k) - \rho^*|$ converges (in any mode mentioned above) to zero with a certain convergence rate, then the EA approaches the target performance value with this rate.

For example, let $\rho(X_k)$ be the best objective function value of the population at generation $k \geq 0$ of a single-criterion EA and ρ^* be the global minimum of the objective function. If Z_k converges to zero then the EA converges to the global minimum. Similarly, let $\rho(X_k)$ be the dominated hypervolume of population X_k and ρ^* the maximal dominated hypervolume in the multi-objective scenario then the population converges to the maximum dominated hypervolume if $Z_k \to 0$ as $k \to \infty$. Other performance concepts can be used accordingly.

3 Analysis

3.1 Rotation of the Coordinate System

Notice that we can rotate the coordinate system by replacing every $x \in \mathbb{R}^n$ by $Tx \in \mathbb{R}^n$ where T is the orthogonal matrix specified in the Theorem 1(d) such that

$$T'AT \;=\; \mathsf{diag}(\nu_1, \ldots, \nu_r, 0, \ldots, 0)$$

for the symmetric, p.s.d. and rank-deficient matrix A of some instance $f(x) = x'Ax$ from our problem class. This leads to

$$f(Tx) = (Tx)'A(Tx) = x'TATx = x'\mathsf{diag}(\nu_1, \ldots, \nu_r, 0, \ldots, 0)\, x = \sum_{i=1}^{r} \nu_i x_i^2$$

with positive eigenvalues $\nu_1, \ldots, \nu_r$ and $r < n$. Since the mutations of our (1+1)-EA are placed spherically symmetric it is sufficient, without loss of generality, to consider only test problems of the form

$$f(x) = \sum_{i=1}^{r} a_i\, x_i^2 \;\to \min! \tag{1}$$

for $x \in \mathbb{R}^n$ with $a_1, \ldots, a_r > 0$ and $r < n$. Evidently, optimal solutions are $x^* \in \mathbb{R}^n$ with $x^* = (0, \ldots, 0, x^*_{r+1}, \ldots, x^*_n)'$ where $x^*_{r+1}, \ldots, x^*_n \in \mathbb{R}$.

3.2 Hessian with Rank 1

At first, we shall consider the extreme case $r = 1$ before we generalize the analysis to arbitrary $r < n$. As a consequence, we can restrict the analysis to the objective function $f(x) = a_1 x_1^2$ with $a_1 > 0$. The objective function value of the mutated individual with parent $x \in \mathbb{R}^n$ is a random variable defined via

$$f(x + \sigma Z) \;=\; a_1 (x_1 + \sigma Z_1)^2 \;=\; a_1 (x_1^2 + 2\sigma x_1 Z_1 + \sigma^2 Z_1^2)$$

where $Z \sim N(0, I_n)$ and $Z_1 \sim N(0,1)$. Setting $\sigma = \gamma x_1$ for some $\gamma > 0$ the equation above changes to

$$f(x + \sigma Z) \;=\; \underbrace{a_1 x_1^2}_{f(x)} + 2\gamma \underbrace{a_1 x_1^2}_{f(x)} Z_1 + \gamma^2 \underbrace{a_1 x_1^2}_{f(x)} Z_1^2 \;=\; f(x) \left(1 + 2\gamma Z_1 + \gamma^2 Z_1^2\right) .$$

Since the $(1+1)$-EA accepts a mutated individual only if it is not worse than the parent the expected function value after mutation and selection is given by

$$\begin{aligned} \mathsf{E}[\min\{f(x), f(x+\sigma Z)\}] &= \mathsf{E}\left[\min\{f(x), f(x)\left(1 + 2\gamma Z_1 + \gamma^2 Z_1^2\right)\}\right] \\ &= f(x) \cdot \mathsf{E}[\min\{1, (1+\gamma Z_1)^2\}] . \end{aligned} \tag{2}$$

Notice that $1 + \gamma Z_1 \sim N(1, \gamma^2)$ so that random variable $V := (1 + \gamma Z_1)^2$ has noncentral χ^2 distribution with 1 degree of freedom and noncentrality parameter $\kappa = 1/\gamma^2$. The cumulative distribution function (c.d.f.) of random variable V can be expressed by the c.d.f. $\Phi(x)$ of the standard normal distribution via

$$F_V(x) = \mathsf{P}\{ (1+\gamma Z_1)^2 \le x \} = \mathsf{P}\{ -\sqrt{x} \le 1 + \gamma Z_1 \le \sqrt{x} \} \;=$$

$$\mathsf{P}\left\{ -\frac{\sqrt{x}+1}{\gamma} \le Z_1 \le \frac{\sqrt{x}-1}{\gamma} \right\} \;=\; \Phi\left(\frac{\sqrt{x}-1}{\gamma}\right) - \Phi\left(-\frac{\sqrt{x}+1}{\gamma}\right)$$

for $x \ge 0$ and $F_V(x) = 0$ otherwise. The probability density function (p.d.f.) of V can be determined from its c.d.f. by derivation w.r.t. x:

$$f_V(x) = \frac{dF_V(x)}{dx} = \frac{1}{2\gamma\sqrt{x}} \left[\varphi\left(\frac{\sqrt{x}-1}{\gamma}\right) + \varphi\left(\frac{\sqrt{x}+1}{\gamma}\right)\right] \cdot 1_{[0,\infty)}(x)$$

where $\varphi(x)$ is the p.d.f. of the standard normal distribution. Notice that $\varphi(x) = \varphi(-x)$ for all $x \in \mathbb{R}^n$. Taking into account the identity

$$\min\{1, v\} = v \cdot 1_{(-\infty,1]}(v) + 1_{(1,\infty)}(v)$$

we are in the position to determine the right factor in equation (2) via

$$\mathsf{E}[\min\{1, (1+\gamma Z_1)^2\}] = \mathsf{E}[\min\{1, V\}] = \mathsf{E}[V \cdot 1_{(-\infty,1]}(V) + 1_{(1,\infty)}(V)] \;=$$

$$\mathsf{E}[V \cdot 1_{(-\infty,1]}(V)] + \mathsf{E}[1_{(1,\infty)}(V)] = \mathsf{E}[V \cdot 1_{(-\infty,1]}(V)] + \mathsf{P}\{V > 1\} \;=$$

$$\int_{-\infty}^{\infty} x \cdot 1_{(-\infty,1]}(x) \cdot f_V(x)\, dx \;+\; \mathsf{P}\{V > 1\} \;=\; \int_0^1 x \cdot f_V(x)\, dx \;+\; \mathsf{P}\{V > 1\} .$$

Notice that

$$\mathsf{P}\{\, V > 1 \,\} = 1 - F_V(1) = \frac{1}{2} + \Phi\left(-\frac{2}{\gamma}\right) .$$

It remains to evaluate the integral

$$\int_0^1 x \cdot f_V(x)\, dx \;=\; \int_0^1 \frac{\sqrt{x}}{2\,\gamma} \left[\varphi\left(\frac{\sqrt{x}-1}{\gamma}\right) + \varphi\left(\frac{\sqrt{x}+1}{\gamma}\right)\right] dx$$

where $\varphi(x)$ is the p.d.f. of the standard normal distribution again. After tedious but straightforward calculations one obtains

$$\int_0^1 x \cdot f_V(x)\, dx \;=\; (\gamma^2+1)\left[\Phi\left(\frac{2}{\gamma}\right) - \frac{1}{2}\right] - \gamma\sqrt{\frac{2}{\pi}}$$

and finally the γ-dependent convergence rate

$$c(\gamma) \;:=\; \mathsf{E}[\,\min\{1, V\}\,] \;=\; \gamma^2\left[\Phi\left(\frac{2}{\gamma}\right) - \frac{1}{2}\right] - \gamma\sqrt{\frac{2}{\pi}} + 1\,. \tag{3}$$

A graphical illustration is given in fig. 1. Numerical minimization of (3) leads to the optimum

$$\gamma^* \approx 0.877 \quad \text{with } c(\gamma^*) \approx 0.676$$

for the special case $r = 1$. Notice that the dimension n of the search space neither affects the optimal convergence rate nor the optimal standard deviation

$$\sigma^* = \gamma^*\, x_1$$

for the mutation vector.

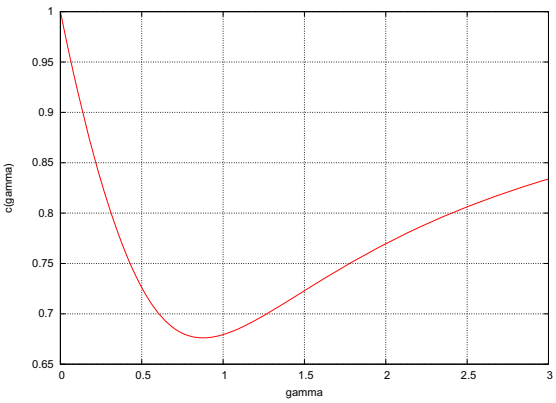


Fig. 1. Convergence rate depending on γ for Hessians with rank 1

Thus, we have proven:

Theorem 2. *The best possible convergence rate of the* (1+1)*-EA with Gaussian mutations for the minimization of a convex quadratic function with rank-deficient Hessian is linear with value* $c^* = 0.676$ *for rank* $r = 1$ *regardless of the dimension of the search space.* □

3.3 Hessian with Rank r < n

The rotation of the coordinate system is not really necessary for the analysis but it was helpful to get a clear view on the essential ingredients of the problem. We now drop the assumption of a properly rotated coordinate system so that the objective function is $f(x) = x'Ax$ for $x \in \mathbb{R}^n$. Since A is assumed to be symmetric, p.s.d. and rank-deficient there exists a (nonunique) Cholesky decomposition $A = B'B$ with upper triangular matrix B [10, p. 13] so that $x'Ax = x'B'Bx = (Bx)'(Bx) = \|Bx\|^2$ where $\|\cdot\|$ denotes the Euclidean norm. The objective function value after mutation is

$$\begin{aligned} f(x + \sigma Z) &= x'Ax + 2\sigma Z'Ax + \sigma^2 Z'AZ \\ &= x'B'Bx + 2\sigma Z'B'Bx + \sigma^2 Z'B'BZ \\ &= (Bx)'(Bx) + 2\sigma (BZ)'(Bx) + \sigma^2 (BZ)'(BZ) \\ &= \|Bx\|^2 + 2\sigma \|BZ\| \cdot \|Bx\| \cdot \cos\omega + \sigma^2 \|BZ\|^2 \end{aligned} \tag{4}$$

where ω is the random angle between BZ and Bx. Notice that ω is independent from $\|BZ\|$ (see [9]) and that $\cos\omega$ has a Beta-distribution with support $(-1, 1)$ and parameters only depending on the rank r (see [16]). Setting $\sigma = \gamma \|Bx\|$ in (4) yields

$$\begin{aligned} f(x + \sigma Z) &= \|Bx\|^2 + 2\gamma \|BZ\| \cdot \|Bx\|^2 \cdot \cos\omega + \gamma^2 \|Bx\|^2 \|BZ\|^2 \\ &= \|Bx\|^2 (1 + 2\gamma \|BZ\| \cdot \cos\omega + \gamma^2 \|BZ\|^2) \\ &= f(x) \cdot (1 + 2\gamma \|BZ\| \cdot \cos\omega + \gamma^2 \|BZ\|^2) \end{aligned} \tag{5}$$

which is the random objective function value after mutation. Since the (1+1)-EA accepts a mutated individual only if it is not worse than the parent the expected function value after mutation and selection is given by

$$\mathsf{E}[\min\{f(x), f(x+\sigma Z)\}] = f(x) \cdot \mathsf{E}[\min\{1, 1+2\gamma \|BZ\| \cdot \cos\omega + \gamma^2 \|BZ\|^2\}] \tag{6}$$

Evidently, $\|BZ\|$ and ω are independent and both do not depend on x (but on r). It follows that the expected convergence rate, i.e., the $\mathsf{E}[\cdot]$ part of (6), is a fixed value for given problem $f(x) = x'Ax$, as A determines B, its eigenvalues and the rank r. This fact reveals a linear convergence rate.

Theorem 3. *The best possible convergence rate of the* (1+1)*-EA with Gaussian mutations for the minimization of a convex quadratic function with rank-deficient Hessian is linear with a value only depending on the positive eigenvalues of the Hessian and its rank, regardless of the dimension of the search space.* □

This result can be extended further. Recall from (1) that the problem on which the (1+1)-EA is operating is nothing more than just a convex quadratic function with full rank in the r-dimensional subspace of $\mathbb{R}^n$. Therefore we can rewrite the existing theory for problems with full rank Hessians and dependence on n to a theory of convex quadratic problems with rank-deficient Hessians by replacing the dimension n of the search space with the dimension r of the spanned subspace (the rank of A). An immediate consequence of this observation is the validity of Jägersküpper's results on self-adaptive EAs on positive definite quadratic forms [11, 12] in the subspace of dimension r.

Theorem 4. *The convergence rate of the* (1+1)*-EA with self-adaptive Gaussian mutations in the spirit of Rechenberg's* 1/5*-success rule for the minimization of a convex quadratic function with rank-deficient Hessian is linear with a value only depending on the success rule parameters and the positive eigenvalues as well as the rank of the Hessian regardless of the dimension of the search space.* □

4 Implications for Multi-Objective Evolutionary Algorithms

The goal of multiobjective optimization is simultaneous minimization of $d \geq 2$ objective functions. In [4] it was proven that in case of $d = 2$ objectives the $(1 + 1)$-SMS-EMOA is algorithmically equivalent to the $(1 + 1)$-EA if the latter minimizes the sum of both objectives. Moreover, if both objective functions are convex quadratic functions where at least one function has a Hessian of full rank then it was shown that the $(1 + 1)$-SMS-EMOA converges geometrically fast to the Pareto front.

With the result of the previous section we can extend the result as follows: Let the convex quadratic objective functions both have rank-deficient Hessians A and B. As a consequence, we have for all $x \in \mathbb{R}^n \setminus \{0\}$

$$f_1(x) + f_2(x) = \underbrace{x'Ax}_{\geq 0} + \underbrace{x'Bx}_{\geq 0} = x'(A + B)x \geq 0$$

with strict inequality if and only if $\mathsf{rank}(A + B) = n$. In this case, the theorem in [4] is also valid. But if $\mathsf{rank}(A + B) < n$ then we may apply Theorem 3 to generalize the result in [4]:

Theorem 5. *The* (1+1)*-SMS-EMOA with self-adaptation applied to a bi-objective optimization problem* $\min\{f : \mathbb{R}^n \to \mathbb{R}^2\}$ *approaches an element of the Pareto front with linear order of convergence if both objective functions are quadratically convex with p.s.d. Hessian matrices.* □

In [5] (section 6) it was proven that in case of $d = 2$ objectives the $(1 + 1)$-SMS-EMOA *with fixed reference point* $r \in \mathbb{R}^2$ is algorithmically equivalent to the $(1 + 1)$-EA if the latter *maximizes* the surrogate function $f^s(x) = [r_1 - f_1(x)] \cdot [r_2 - f_2(x)]$. Moreover, if the objective functions are linear, i.e., $f(x) = (a_0 + a'x, b_0 + b'x)'$, then $-f^s(x)$ is convex provided that $-ab'$ is positive semidefinite.

In this case the theorem in [5] guaranteed a linear convergence rate between 0 and some $c \in (0, 1)$. Before improving the result we figure out under which conditions we can assert that $-ab'$ is p.s.d. for sure.

Theorem 6. *Let $a \in \mathbb{R}^n$ with $n > 1$, $a \neq 0$ and $A = a\,a'$.*
a) A is symmetric.
b) A is p.s.d.
c) $\mathsf{rank}\ A = 1$.
d) $\nu_1 = a'a > 0$.

Proof.
a) $a_{ij} = a_i \cdot a_j = a_j \cdot a_i = a_{ji}$ for all $i, j = 1, \ldots, n$.
b) $x'(a\,a')x = (x'a)\,(a'x) = (a'x)\,(a'x) = (a'x)^2 \geq 0$ for all $x \in \mathbb{R}^n$.
c) By construction, the ith row is a multiple of the first row for $i = 2, \ldots, n$.
d) Since $\mathsf{trace}\ A = \sum_{i=1}^n \nu_i$ and $\mathsf{trace}\ A = \mathsf{trace}\ aa' = \sum_{i=1}^n a_i^2 = a'a > 0$ we know that $\sum_{i=1}^n \nu_i = a'a > 0$. Owing to Theorem 1(c) and Theorem 6(c) we have $\nu_1 > 0$ and $\nu_i = 0$ for $i = 2, \ldots, n$. Thus, $\mathsf{trace}\ aa' = \nu_1 = a'a > 0$. □

Thus, if $b = -a$ then the preconditions of Theorem 2 are fulfilled and we may state:

Theorem 7. *The $(1+1)$-SMS-EMOA with fixed reference point and spherically symmetric mutations $Z \sim N(0, \sigma_t^2\, I_n)$ maximizes the dominated hypervolume of the linear problem*

$$f(x) = (a'x + a_0, b'x + b_0)' \to \min!, \quad x \in \mathbb{R}^n$$

with linear convergence rate if $b = -a$. □

5 Conclusions

The determination of the convergence rates of simple evolutionary algorithms for quadratic convex functions can be extended from the case of positive definite to positive semidefinite Hessians, or equivalently, from fully ranked to rank-deficient Hessians: The order of the convergence rate is not affected. The analysis also revealed that optimizing rank-deficient quadratic problems can be treated as optimizing positive definite quadratic problems in subspaces of dimension $r < n$, where r is the rank of the Hessian. Moreover, the convergence theory for EAs in $\mathbb{R}^n$ for quadratic problems actually does not depend on the dimension n of the search space but on the codimension r of the Hessian's kernel, i.e., the rank of the Hessian. Therefore the broadly accepted tenet that the best convergence rate depends on the problem dimension has to be adapted: the best convergence rate depends on the dimension of the dimensionally largest subspace affecting the objective function value. Finally, these results immediately extend recent convergence result for the $(1 + 1)$-SMS-EMOA used in bicriteria optimization.

References

1. Abadir, K., Magnus, J.: Matrix Algebra. Cambridge Univ. Press, New York (2005)
2. Bäck, T.: Order statistics for convergence velocity analysis of simplified evolutionary algorithms. In: Whitley, L.D., Vose, M.D. (eds.) Foundations on Genetic Algorithms, vol. 3, pp. 91–102. Morgan Kaufmann, San Fransisco (1995)
3. Bäck, T., Rudolph, G., Schwefel, H.P.: Evolutionary programming and evolution strategies: Similarities and differences. In: Fogel, D.B., Atmar, W. (eds.) Proc. of the 2nd Annual Conf. on Evolutionary Programming, pp. 11–22. Evolutionary Programming Society, La Jolla (1993)
4. Beume, N., Laumanns, M., Rudolph, G.: Convergence Rates of (1+1) Evolutionary Multiobjective Optimization Algorithms. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6238, pp. 597–606. Springer, Heidelberg (2010)
5. Beume, N., Laumanns, M., Rudolph, G.: Convergence rates of SMS-EMOA on continuous bi-objective problem classes. In: Beyer, H.G., Langdon, W. (eds.) Proceedings of the 11th Int'l Conf. on Foundations of Genetic Algorithms (FOGA XI), pp. 243–251. ACM Press, New York (2011)
6. Beyer, H.G.: Toward a theory of evolution strategies: Some asymptotical results from the $(1 \dot{+} \lambda)$–theory. Evolutionary Computation 1(2), 165–188 (1993)
7. Beyer, H.G.: Toward a theory of evolution strategies: The (μ, λ)–theory. Evolutionary Computation 2(4), 381–407 (1994)
8. Beyer, H.G.: The Theory of Evolution Strategies. Springer, Heidelberg (2001)
9. Fang, K.T., Kotz, S., Ng, K.W.: Symmetric Multivariate and Related Distributions. Chapman and Hall, London (1990)
10. Householder, A.S.: The Theory of Matrices in Numerical Analysis, corrected edn. Dover Books, New York (1975)
11. Jägersküpper, J.: How the (1+1) ES using isotropic mutations minimizes positive definite quadratic forms. Theoretical Computer Science 361(1), 38–56 (2006)
12. Jägersküpper, J.: Algorithmic analysis of a basic evolutionary algorithm for continuous optimization. Theoretical Computer Science 379(3), 329–347 (2007)
13. Oyman, A.I., Beyer, H.G.: Analysis of the $(\mu/\mu, \lambda)$-ES on the parabolic ridge. Evolutionary Computation 8(3), 267–289 (2000)
14. Oyman, A.I., Beyer, H.G., Schwefel, H.P.: Analysis of the $(1, \lambda)$-ES on the parabolic ridge. Evolutionary Computation 8(3), 249–266 (2000)
15. Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. Frommann–Holzboog, Stuttgart (1973)
16. Rudolph, G.: Convergence Properties of Evolutionary Algorithms. Kovač, Hamburg (1997)
17. Rudolph, G.: Convergence rates of evolutionary algorithms for a class of convex objective functions. Control and Cybernetics 26(3), 375–390 (1997)
18. Rudolph, G.: Asymptotical Convergence Rates of Simple Evolutionary Algorithms under Factorizing Mutation Distributions. In: Hao, J.-K., Lutton, E., Ronald, E., Schoenauer, M., Snyers, D., et al. (eds.) AE 1997. LNCS, vol. 1363, pp. 223–233. Springer, Heidelberg (1998)
19. Rudolph, G.: Local convergence rates of simple evolutionary algorithms with Cauchy mutations. IEEE Transactions on Evolutionary Computation 1(4), 249–258 (1998)
20. Scheel, A.: Beitrag zur Theorie der Evolutionsstrategie. Doctoral Dissertation, TU Berlin, Berlin (1985)
21. Schwefel, H.P.: Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie. Birkhäuser, Basel (1977)

The Scale-Up Performance of Genetic Algorithms Applied to Group Decision Making Problems

Tatiana Tambouratzis and Vassileios Kanellidis

Department of Industrial Management and Technology,
University of Piraeus, Piraeus, Greece
tatianatambouratzis@gmail.com
bcanell@unipi.gr

Abstract. The scale-up performance of genetic algorithms applied to group decision making problems is investigated. Ordinal intervals are used for expressing the individual preferences of the decision makers, as provided independently for each course of action. Genetic algorithms have been found capable of swiftly returning optimal ranking solutions, with computational complexity (the relationship between the number of available courses of action and the number of generations until convergence) expressed by a fourth order polynomial, but found practically independent of the number of decision makers.

Keywords: Group decision making. decision makers, courses of action, preference ranking, ordinal interval numbers, genetic algorithms, scaling.

1 Introduction

Group decision making (GDM) [1-2] selects the preferred course of action (COA) out of a number of available COAs. A group of decision makers (DMs) is employed to this end, with each DM expressing his/her preference for the COAs, following which the preferences are aggregated and the collectively preferred COA is determined.

A means of implementing GDM problems is preference ranking via ordinal interval numbers. Preference ranking GDM via ordinal interval numbers produces an ordering of all the COAs in terms of collective preference, thereby showing a significant advantage over classic GDM in terms of robustness as well as efficiency in the case that the overall preferred COA cannot be implemented and an alternative COA must be selected (namely the COA ranked second, third etc. best until an implementable COA is found).

In this piece of research, the scale-up performance of genetic algorithms (GAs) [3] applied to preference ranking GDM via ordinal interval numbers is investigated for increasing numbers of available COAs and/or DMs. GAs have been found capable of swiftly returning optimal ranking solutions, with computational complexity (the relationship between the number of available courses of action and the number of generations until convergence) expressed by a fourth order polynomial, but found practically independent of the number of decision makers. An example taken from the relevant literature is used to demonstrate these points.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 161–168, 2013.

The presentation is organized as follows: Sections 2 and 3 introduce preference ranking GDM via ordinal interval numbers and GAs, respectively; Section 4 describes the construction and operation of the GA for efficiently producing optimal (i.e. collectively preferred) orderings of the COAs, while Section 5 evaluates the scale-up potential of GAs using a large-scale GDM problem found in the literature; finally, Section 6 concludes the paper.

2 Preference Ranking GDM via Ordinal Interval Numbers

In preference ranking GDM via ordinal interval numbers, the m DMs independently express their preferences concerning the ranking of the n COAs via ordinal interval numbers of the form [a b] ($1 \le a \le b \le n$), specifying that a COA is ranked either ath, a+1th, …, b-1th, or bth best in a DM's preferences. The relation between the values of a and b determines the following:

- if $1 \le a$ and $b < n$, *or* $1 < a$ and $b \le n$, it is possible to consider (a) all the numbers in the ordinal interval number as equivalent in the DM's preference rankings, or (b) the nearer/further the numbers are to the centre of the ordinal interval number, the greater/smaller the DM's preference ranking[1];
- if $a=b$, a trivial (single number) ordinal interval number is created specifying that the COA is ranked exactly a^{th} best in the DM's preferences;
- if $1=a$ and $b=n$, the DM expresses no preference ranking for the COA, which this is equivalent to the DM providing no preference for the COA.

In this piece of research, purely binary reward-penalty assignments are employed, with all the numbers in the ordinal interval number being ranked equal in terms of reward/penalty. Assuming that ordinal interval numbers [x_{cd} y_{cd}] are provided by the dth ($1 \le d \le m$) DM for the cth ($1 \le c \le n$) COA, and that a given ranking $v=\{v_c\}$ ($1 \le c \le n$) of the COAs exists, the binary reward-penalty assignment allots rewards of +1 if v_c falls within [x_{cd} y_{cd}], penalties of -1 if v_c falls outside [x_{cd} y_{cd}], and 0 if the DM provides no preference. Such an assignment implies that the optimal ranking of the COAs is the one that collectively satisfies the maximum number of ordinal interval numbers.

It is useful to assign importance values imp_d ($1 \le d \le m$) to the DMs, expressing each DM's relative familiarity/expertise with the problem. By convention

$$\sum_{d=1}^{m} imp_d = 1 . \tag{1}$$

with $imp_d = \frac{1}{m}$ ($1 \le d \le m$) if the same confidence is assigned to all of the DMs' ordinal interval numbers.

[1] The first representation is used here.

3 GAs

GAs derive their inspiration from the evolution of species observed in nature, as this may occur at different time scales (from hours, weeks, or months to years or even millennia).

GAs operate by manipulating populations of candidate solutions to a problem. Each candidate solution is encoded as a chromosome, although ensembles of chromosome may also be used for expressing a candidate solution. Every chromosome comprises genes, with every gene representing an elementary component of the problem. The different candidate solutions are expressed via distinct combinations of values of the constituent genes. A fitness function is employed which evaluates the quality of any given candidate solution in constituting a solution to the problem.

GA operation begins with a random population of chromosomes (candidate solutions) which evolves via the following three operators:

(a) crossover, i.e. the random exchange of genes between two or more chromosomes,

(b) mutation, i.e. the random and with small probability change in the values of individual genes of a chromosome, and

(c) selection, i.e. the probability of including a chromosome in the evolving population constituting a function of its quality (fitness) in representing a solution to the problem.

The repeated application of these operators promotes the inclusion of chromosomes of progressively high fitness in the population, and thus drives the population towards increasing average fitness. As a result, and after an adequate number of generations, a (near-)optimal solution to the problem emerges from the population.

4 GAs for Preference Ranking GDM via Ordinal Interval Numbers

4.1 Chromosome Construction

A given chromosome *CH* comprises n genes $g=\{g_c\}$ $(1\leq c\leq n)$, where the cth gene g_c represents the cth COA and takes on a distinct integer value between 1 and n specifying the rank assigned to the cth COA according to the Borda-Kendall representation [4-5][2]. Ties in ordering the COAs are not permitted, whereby the set of values of the genes $\{g_c\}$ $(1\leq c\leq n)$ must constitute a permutation of numbers 1 to n.

[2] For instance, for a given chromosome {4 2 1 3}, the first COA is ranked fourth, the second COA is ranked second, the third COA is ranked first, and the fourth COA is ranked third. This chromosome represents the relationship $S_3 > S_2 > S_4 > S_1$, where $A > B$ denotes that A is ranked higher than B in terms of preference.

4.2 Fitness Function

The degree to which chromosome *CH* satisfies the DMs' ordinal interval numbers is expressed by the algebraic sum of the rewards and penalties calculated according to the binary reward-penalty assignment via the inclusion (or not) of the ranking of the *c*th ($1 \le c \le n$) COA - represented by the value of the *c*th gene g_c ($1 \le c \le n$) - in the corresponding ordinal interval number of the *d*th DM ($1 \le d \le m$). Each contribution is scaled by the importance value assigned to the respective DM, resulting in:

$$f_c = \sum_{d=1}^{m} imp_d . rp_{cd} \qquad (1 \le c \le n). \tag{2}$$

The fitness value fit_{CH} of chromosome *CH* constitutes the sum of the contributions f_c ($1 \le c \le n$)

$$fit_{CH} = \frac{1}{n} \cdot \sum_{c=1}^{n} f_c \tag{3}$$

where the scaling factor $\frac{1}{n}$ ensures that the fitness value fit_{CH} ranges within [-1 +1].

Since the total number *n*! of chromosomes that can be exhaustively generated by permuting numbers 1, 2, 3, …, *n*, becomes unmanageable for values of *n* larger than 4 or 5 (as does the evaluation of their fitness values), GAs constitute a viable choice for producing an optimal solution (i.e. .a collectively preferred COA ranking that maximally satisfies the DMs' ordinal interval numbers for the given COAs) after a small - yet sufficient - number of generations.

4.3 GA Operation

Numerous crossover/mutation operators and selection methods have been proposed for problems of permutation. For the GDM problem tackles here, numerous combinations of the most popular crossover/mutation operators (like partially patched crossover, cycle crossover, order-based crossover, Stefan Jakobs Crossover or SJX) and selection methods (like proportional selection, tournament selection, random pairing) have been tested. The combination that has been found to perform quite well consists of the SJX [6] methodology as the crossover operator, roulette-wheel as the selection operator and gene-swapping as a the mutation operator.

The SJX crossover ensures that the two chromosomes resulting from mixing the genes of the two original chromosomes are valid, i.e. that their rankings (values of their genes) continue to constitute permutations of the numbers 1, 2, …, *n*; for preference ranking GDM via ordinal interval numbers, the optimal crossover rate equals 0.9.

The mutation operator simply swaps the location of two randomly selected genes of each chromosome, thus again preserving validity of the mutated chromosomes; a mutation probability of 0.01 is used in the following.

Selection employs the principle of "roulette-wheel selection". For each generation, selection from the current population is performed P times, where P is the size of the population. In order not to give chromosomes of higher fitness too great of an advantage over chromosomes of lower fitness during the selection procedure – and to, thus, avoid premature convergence to a sub-optimal solution), the fitness scaling technique of rank scaling is employed, namely the chromosomes are scored according to their ranks (i.e. their positions in the list of chromosomes sorted in descending order of their fitness values) rather than on its actual fitness values.

Finally, termination of GA operation occurs as soon as no change is observed in the fitness value of the fittest chromosome of the population for 500 consequent generations.

5 Demonstration - Results

The application of GAs to preference ranking GDM via ordinal interval numbers is demonstrated on a large-scale (six-DM nine-COA) problem that has already been investigated in [7-8]. As shown in Table 1, the six DMs (n=6) provide their preference rankings for the nine COAs (n=9) via trivial (single-valued) ordinal interval numbers; ties are allowed, whereby more than one COA may be assigned the same ranking by a DM. In the following, all DMs are assumed equivalent in terms of expertise, whereby they are assigned uniform importance values of 1/5, and the GA population invariably (for all the values of n investigated) comprises 50 chromosomes.

Table 1. The Six-DM Nine-COA ordinal interval number GDM problem of [7-8]

COA **DM**	1	2	3	4	5	6	7	8	9
1	4	9	2	1	5	6	3	8	7
2	1	7	2	8	9	6	5	4	3
3	7	9	5	1	2	8	3	4	6
4	1	4	2	5	3	8	9	6	7
5	2	7	1	4	3	9	5	6	8
6	1	9	2	7	4	7	4	4	3

The scale-up potential of the proposed approach is investigated for n=5, 6, 7, 8, 9 COAs. This is implemented by beginning from the original six-DM nine-COA problem shown in Table 1 and progressively ignoring the last COA (9^{th}, 8^{th}, 7^{th}, and 6^{th}) and limiting the remaining ordinal interval numbers such that they do not to exceed the current value of n. Table 2 shows the scale-up results of the GA (for the different values of n), expressed in terms of

(a) the total number of combinatorially possible rankings;
(b) .the highest fitness value determined analytically as well as via the proposed GA;
(c) the accuracy of the GA in terms of the frequency of reaching an optimal ordering of the COAs;
(d) the efficiency of the GA, namely the mean number of generations required until GA convergence is achieved[3];
(e) the total number of chromosomes generated until GA convergence; this number –provides a rough estimate of the amount of problem space visited during GA operation.

The following points are made:

(a) The first row of Table 2 shows that the process of producing, and subsequently, evaluating all the COA rankings quickly becomes inefficient (time- as well as effort-consuming) for increasing values of n.
(b) A comparison between the second and third rows of Table 2 illustrates the successful (accurate as well as efficient) operation of the GA: the fitness value of the optimal chromosome(s) (i.e. COA ranking(s)) coincide when evaluated analytically as well as via the GA. When observing the fourth row of the same Table, however, it can be seen that GA accuracy falls significantly for n=9, a fact that is due to the small population of 50 chromosomes used for exploring the vast problem space of 362,880 combinatorially possible solutions via the GA. It is interesting that a population of 100 chromosomes produced significantly higher accuracy without compromising efficiency, suggesting that the size of the GA population can vary as a function of n.

Further investigating efficiency, a 4^{th} degree polynomial has been found sufficiently accurate (error of the order of e^{-10} for the best-fit in a least-squares sense) in approximating the relationship between the value of n and the mean number of generations (over 100 trials) until convergence.

The last row of the Table shows that the number of chromosomes created until convergence exceeds the total number of combinatorially possible solutions for n=5 and n=6. This suggests that evaluation of the relatively "few" (120 and 720, respectively) fitness values of the combinatorially possible COA orderings is preferable to employing the GA. For higher values of n, however, the total number of chromosomes created by the GA becomes a fraction of the total number of combinatorially possible solutions (32%, 15%, and only 4% for n=7, 8, and 9, respectively), highlighting the scale-up potential of the GA. Although this fraction is underestimated in that it does not take into account the re-appearance of the fitter chromosomes in the evolving population of chromosomes, it shows that GA

[3] Convergence occurs as soon as the highest fitness value of the chromosomes in the evolving population does not change for 500 generations; the first generation at which convergence is shown in Table 2.

convergence is swift and focused early on during GA operation in promising areas of the problem space. The fact that only 4% of the population (at best) is investigated for n=9 further supports the increase in population size proportionally to the value of n.

Table 2. Scale-up performance characteristics of the GA applied to the six-DM nine-COA GDM problem

***n* value** **problem characteristics**	**5**	**6**	**7**	**8**	**9**
number of combinatorially possible rankings	120	720	5,040	40,320	362,880
highest fitness (analytical)	0.1333	0	-0.0952	-0.2083	-0.1852
highest fitness (GA)	0.1333	0	-0.0952	-0.2083	-0.1852
accuracy (% correct Solutions)	100	99	100	100	41
mean number of generations until convergence	3	42	32	119	302
number of chromosomes until convergence	150	2,100	1,600	5,950	15,100

Some final points are mentioned next:

- The value of m does not affect the accuracy of GA operation, as it is not implicated in the evaluation of the fitness function or of GA operation and convergence; by appearing in Equ. (2), it only weakly affects computational complexity.
- A decreasing trend of the highest fitness value is observed for rising values of n. This is not unexpected, as it becomes increasingly difficult to accommodate for the larger number of constraints that need to be satisfied for higher values of n.
- A comparison of the GA approach with the linear programming GDM methodologies of [7-8] highlights the transparent, constraint-free means of expressing the GDM problem, the computationally efficient means of converging towards an optimal or near-optimal) solution, and the robustness of the proposed approach to deal with ties in the DMs' preferences.

6 Conclusions

Genetic algorithms have been found to be appealing alternatives (in terms of both accuracy and efficiency) to existing linear programming methodologies, when applied to group decision making problems expressed as preference ranking via ordinal interval numbers. The proposed representation and search for an optimal solution is independent of the existence of ties in the preferences of the decision makers. Computational complexity is expressed by a fourth order polynomial, but found practically independent of the number of decision makers. An example appearing in the relevant literature confirms these points, while also suggesting that a gradual increase in population size for group decision problems involving more alternative courses of action would increase accuracy.

References

1. Moore, C.M.: Group Techniques for Idea Building (Applied Social Research Methods). Sage Publications Inc., Thousand Oaks (1987)
2. Hwang, C.L., Lin, M.J.: Group Decision Making under Multiple Criteria: Methods and Applications. Springer, Berlin (1987)
3. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison Wesley (1989)
4. Borda, J.C.: Memoire sur les Elections an Scrutin, Histoire de l' Academie Royale de Science, Paris (1784)
5. Kendall, M.: Rank Correction Methods, 3rd edn. Hafners, New York (1962)
6. Jakobs, S.: On genetic algorithms for the packing of polygons. European Journal of Operational Research 88, 165–181 (1996)
7. Islei, G., Lockett, G.: Group decision making: suppositions and practice. Socio-Economic Planning Sciences 25, 67–82 (1991)
8. Wang, Y.M., Yang, J.B., Xu, D.L.: A preference aggregation method through the estimation of utility intervals. Computers & Operations Research 32, 2027–2049 (2005)

Using Genetic Programming to Estimate Performance of Computational Intelligence Models

Jakub Šmíd[1] and Roman Neruda[2]

[1] Charles University in Prague, Faculty of Mathematics and Physics, Malostranské nám. 2/25, Prague, Czech Republic
[2] Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod Vodárenskou víží 2, 18207 Prague, Czech Republic

Abstract. This paper deals with the problem of choosing the most suitable model for a new data mining task. The metric is proposed on the data mining tasks space, and similar tasks are identified based on this metric. A function estimating models performance on the new task from both the time and error point of view is evolved by means of genetic programming. The approach is verified on data containing results of several hundred thousands machine learning experiments.

Keywords: metalearning, genetic programming, data mining, performance prediction.

1 Introduction

Data mining – the process of finding new patterns in data sets – is now widely used in medicine, economics, bioinformatics and other important areas of human interest. Many different algorithms exist and are used for this task of pattern extraction. According to the no free lunch theorem [1], the average performances of data mining algorithms on all data mining tasks are equal. That means that superior performance of any algorithm over one class of tasks is paid for in performance over another class. Therefore, the key to success when dealing with a data mining task is in binding the task with an algorithm having superior performance on the class of similar tasks.

Data mining tasks have many parameters (task type, number of instances, number of attributes, types of attributes, etc.), and they may be compared based on these parameters. One can assume that similar tasks belong to the same class, and that the performance of any algorithm will be similar to the tasks of that class (that means tasks with similar parameters). Having a new data mining task, the estimated performance of all algorithms in question may be calculated by utilizing performance of those algorithms on tasks similar with the one at hand. The algorithms with the best estimation are suggested as candidates for solving the task in question.

This paper proposes a metalearning method that estimates performance of algorithms on a given data mining task. The metalearning studies how learning

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 169–178, 2013.

systems can increase their efficiency through experience [2]. It differs from base-learning in the scope of the level of adaptation; whereas learning at the base-level is focused on accumulating experience on a specific learning task, learning at the meta-level is concerned with accumulating experience on the performance of multiple applications of a learning system. Briefly stated, the field of metalearning is focused on the relation between tasks or domains and learning strategies.

Our method utilizes the genetic programming (GP) approach to develop a meta-model which is implemented in the form of a software agent exposing an interface which returns evaluation results in real time. It has become an integral part of our data mining multi-agent system and it serves as a recommending service for selecting the best machine learning model based on previous experience.

2 Related Work

Authors of [3] employ the k-nearest neighbor algorithm with a distance function based on a set of statistical, information theoretic, and other dataset characterization measures in order to identify the set of similar already computed tasks (the so-called *zooming* phase). In the second phase (called *ranking*), a ranking on the basis of the performance information of the candidate algorithms on the selected datasets is constructed. The adjusted ratio of ratios ranking method is presented which processes performance information based on accuracy and time.

The relevance of the processed dataset d_i to the dataset d_j at hand is defined in terms of similarity between them, according to meta-attributes. It is given by a function:

$$dist\left(d_i, d_j\right) = \sum\nolimits_x \delta\left(v_{x,d_i}, v_{x,d_j}\right), \tag{1}$$

where d_i and d_j are datasets, v_{x,d_i} is the value of meta-attribute x for dataset d_i, and $\delta\left(v_{x,d_i}, v_{x,d_j}\right)$ is the distance between the values of meta-attribute x for datasets d_i and d_j. All meta-attributes were normalized.

The k-nearest neighbor algorithm is then used to identify k cases nearest to the dataset in hand.

The adjusted ratio of ratios uses information about accuracy and execution time to rank the given classification algorithms. The auxiliary term $ARR^{d_i}_{a_p,a_q}$ is defined as:

$$ARR^{d_i}_{a_p,a_q} = \frac{\frac{SR^{d_i}_{a_p}}{SR^{d_i}_{a_q}}}{1 + \frac{\log\left(\frac{T^{d_i}_{a_p}}{T^{d_i}_{a_q}}\right)}{K_T}}, \tag{2}$$

where $SR^{d_i}_{a_p}$ and $T^{d_i}_{a_p}$ are the success rate and duration of algorithm a_p on the dataset d_i, and K_T is a user-defined value that determines the relative importance of time.

Finally, we derive the overall mean adjusted ratio of ratios for each algorithm:

$$ARR_{a_p} = \frac{1}{m-1}\left(\sum\nolimits_{a_q} \frac{\sum_{d_i} ARR^{d_i}_{a_p,a_q}}{n}\right), \tag{3}$$

where m is the number of algorithms, n is the number of datasets. The ranking is derived directly from this measure.

Authors of [4] take similar approach. During the zooming phase, metadata containing the following information about the task were utilized:

- *Number of attributes* in data,
- *Number of instances*,
- *Data type* (one of the following – categorical, integer, multivariate),
- *Default task type* (set by the user, the most common types are classification and regression),
- *Missing values* (flag whether data contains unknown or unspecified values).

The following metric is defined between two tasks based on their metadata:

$$d\left(m_1, m_2\right) = \sum_{i=1}^{n} w_i d_i\left(m_1\left[i\right], m_2\left[i\right]\right), \tag{4}$$

where m_1, m_2 are metadata of compared tasks, w_i is a weight of each attribute and d_i is distance between metadata attribute i which depends on the type of the attribute. The following formula was used to compute the distance of Boolean and categorical attributes:

$$d_i\left(v_1, v_2\right) = \begin{cases} 0;\ if\ v_1 = v_2 \\ 1;\ otherwise \end{cases}. \tag{5}$$

Missing values are handled as an extra Boolean attribute. The variables v_1 and v_2 represent actual values of the attribute i.

To compute the distance of numerical attributes, the following formula that maps two values v_1 and v_2 on the interval $\langle 0, 1\rangle$, was used:

$$d_i\left(v_1, v_2\right) = \frac{|v_1 - v_2|}{\max\limits_{v \in i} v - \min\limits_{v \in i} v}. \tag{6}$$

The suggested model is the one with the best performance on the nearest task.

Authors of [5] propose a practical model of Evolutionary Program-induction Algorithms (EPAs) including Genetic Programming. The model corresponds to the following equation:

$$P(\mathbf{t}) \approx a_0 + \sum_{\mathbf{p} \in S} a_{\mathbf{p}} d(\mathbf{p}, \mathbf{t}), \tag{7}$$

where a_i are coefficients, $P(\mathbf{t})$ is a performance of an EPA on the target functionality $\mathbf{t}$, S is a subset of a program search space and d is a function of similarity between the output of the EPA and the target functionality $\mathbf{t}$. Paper deals with the issue of determining the suitable coefficients and the suitable subset of the search space. The model is tested on various tasks.

3 Metalearning Algorithm Design

This section describes our design of a genetic programming based metalearning procedure. First, the metadata and the induced metrics are described, then the genetic programming algorithm is described. Metadata values were extracted from all instances of all data mining tasks computed by our models. Metadata is divided into two categories:

- Task related – all metadata available for each task:
  - Number of instances.
  - Number of attributes.
- Attribute related – all metadata available for each attribute of the task:
  - Type: determines the nature of the attribute and it has one of the following values – real, integer, categorical, Boolean. Additional availability of metadata depends on the type.
  - Missing values: determines whether an attribute can have an unknown value or not.

Brief summary of metadata is shown in the Fig. 1:

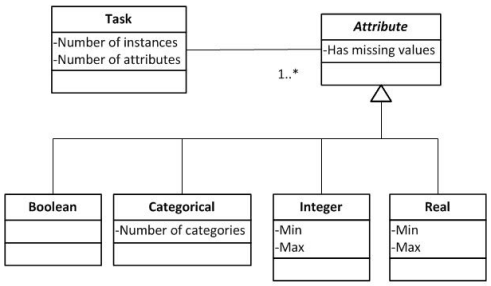


Fig. 1. Available metadata

Our metric proposal is based on the metric proposed in [4]:

$$d\left(m_1, m_2\right) = \sum_{i=1}^{n} w_i d_i\left(m_1\left[i\right], m_2\left[i\right]\right), \tag{8}$$

where m_1, m_2 are metadata of compared tasks, w_i is a weight of each attribute and d_i is distance between metadata attribute i.

This metric yields different results for various orderings of metadata. Even the same data mining tasks, only with different attribute ordering, may be evaluated as distant. We propose a metric that eliminates this issue:

$$d\left(m_1, m_2\right) = d_{categorical}\left(m_1, m_2\right) + d_{integer}\left(m_1, m_2\right) + d_{real}\left(m_1, m_2\right). \tag{9}$$

For the purposes of the metric, Boolean attributes will be treated as categorical attributes with two categories $True$ and $False$. Let min be a minimum value of integer occurring in the metadata, likewise let max be a maximum value of integer occurring in the metadata. Let Σ be an alphabet of integers between min and max. A *string* S over an alphabet Σ is a (finite) concatenation of symbols from S. The *length* of a string S is the number of symbols in S, denoted by $|S|$. $S[i]$ denotes the i^{th} symbol of S. *Alignment* of two strings S_1 and S_2 (without loss of generality, let us assume that $|S_1| \geq |S_2|$) is a function f that for every position in S_1 returns a position in S_2, or a special symbol GAP. In addition, for every position j in S_2, there must exist exactly one position i in S_1 such that $f(i) = j$.

Let the *string representation* of categorical attributes be a concatenation of the number of categories of categorical attributes. Let $C_2\,[GAP]$ be treated as 0. Let C_1,C_2 be string representations of categorical attributes of metadata m_1, m_2. Then we define the categorical distance as:

$$d_{categorical}\,(m_1, m_2) = \min_{f} \sum_{i=1}^{|C_1|} |C_1\,[i] - C_2\,[f\,(i)]|\,. \tag{10}$$

An algorithm computing $d_{categorical}$ with the complexity of $O\,(n \log\,(|C_1|))$ was introduced in our previous work [6].

The term $d_{integer}$ is defined the same way, except that the difference between minimum and maximum value of the attribute is used instead of a number of categories.

We have proposed d_{real} as:

$$|R_1 - R_2| \max\,(4, \{|C_1[i] - C_2[f(i)]||i \in F\}\,, \{|I_1[i] - I_2[l(i)]||i \in F\}), \tag{11}$$

where R_1, R_2 are numbers of real attributes in m_1, m_2, f is the alignment used in $d_{categorical}$, $F = dom(f)$, and l is the alignment used in $d_{integer}$. We argue that real attributes do not depend on its Min and Max (because even if Min and Max are arbitrarily near, there is still infinity of different points between them) and should be the most significant. Definition of the real distance does not depend on the order of real, integer and categorical attributes. It would be undesirable if there were other optimal $f^{'}$, $l^{'}$ alignments that would be more optimal in the sense of the real distance compared to f, l. We have shown in [6] that this is not the case.

Estimation functions were evolved by the genetic programming and were represented by two trees: one for estimating accuracy and another one for estimating time consumption. Each estimation function receives as its inputs a task to estimate, task metadata, a model to estimate, and performance results of the model on similar tasks based on the metric proposed in the previous section. Therefore, we need to propose two domains for the GP algorithm. If not said otherwise, the proposed functions and terminals can be used regardless of the type of program evolved. All functions and terminals are type consistent, which means that all functions have all arguments and output of the same type, a real number in our

case. If some n-ary function is not defined on the whole R^n, its extension on the whole R^n is used instead.

For the sake of simplicity, only functions representing elementary operations, square root, logarithm and Boolean functions lesser than, lesser than or equal are used.

Constant terminals: we have proposed terminals that represent real and integer numbers. When creating such a terminal, a random number is generated and set as a value of the new terminal.

Previous results terminals: terminals that represent previous results of the model in question were proposed. When creating such a terminal a random number i is generated. The value of the new terminal is the root mean squared error of the model on the i^{th} nearest task for the accuracy domain and computation duration of the i^{th} nearest task for the time domain. To measure the time elapsed, the computation duration was the only option. Computation duration is the time interval between start and finish of the computation and it is expressed in seconds.

Metadata terminals: A terminal for each type of metadata available can be considered, but this amount of terminals is not necessary for estimating the duration or accuracy. Instead, the following terminal that aggregates metadata information of the task on an input is proposed:

$$terminal\ complexity = \sum_{a_i} \delta\left(a_i\right) \log(Task.NumberOfInstances), \tag{12}$$

where a_i is an i^{th} attribute of the task, $\delta\left(a_i\right)$ is the complexity of an attribute defined as:

$$\delta\left(a_i\right) = \begin{cases} 1;\ if\ a_i\ is\ Boolean\ or\ categorical\ attribute \\ 2;\ if\ a_i\ is\ integer\ attribute \\ 3;\ if\ a_i\ is\ real\ attribute \end{cases} . \tag{13}$$

and Task.NumberOfInstances is the number of instances of the task on input. The purpose of this terminal is to represent the task complexity. Integer attributes are nearly always harder to compute than Boolean and categorical attributes, and similarly real attributes are nearly always harder to compute than integer attributes. The complexity of the task also rises with a logarithm of the number of instances.

The estimation agent architecture consists of 4 parts – the *input part*, the *zooming part*, the estimation part and the output part. The input part receives queries and forwards them to the zooming part. The zooming part handles zooming and forwards the results to the estimation part. Another role of the zooming part is to obtain the data from both storages. The *estimation part* encapsulates estimation trees. Its goal is to return a time and accuracy estimation of all models that occur in Computation results storage on the task in the query. The *output part* handles the estimation results delivery to the inquirer. All parts and their purposes are overviewed in the Fig. 2:

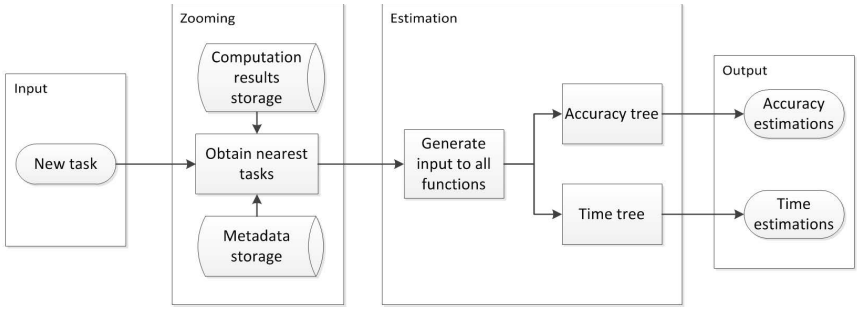


Fig. 2. Architecture overview

For each model in the Computation results storage, the estimation is as following:

- In zooming, we first identify several closest tasks already computed by the model based on their metadata and metrics proposed. For this process, we use the actual states of Computation results and metadata storage. The number of tasks obtained by zooming depends on the GP trees evolved, to be more precise, on the highest argument of the Result terminal used in the trees.
- In estimation, the input task and the tasks obtained by zooming are used to compute an input for all terminals in the generated trees. The trees are evaluated and their output is returned.

4 Experiments

We had 109 732 records of previous computation results at our disposal for training of our models. The validation set was extracted from 649 964 previously unseen performance records. The tasks computed were commonly known tasks from [7], while the computational models correspond to several Weka [8] methods, such as multilayer perceptron and RBF networks, and similar. The large number of records contains each pair of task and model with different parameter settings. 50 GP runs were performed for both duration and accuracy experiments. The fitness progression of the best runs is shown in the Fig. 3. The following abbreviations are used for models in the figures showing validation results (Figs. 4 and 5): MLP is a multilayered perceptron, RND is a random tree algorithm, RBF stands for a RBF network, J48 and PART are decision tree algorithms, NNge is the nearest neighbor like algorithm using non-nested generalized exemplars.

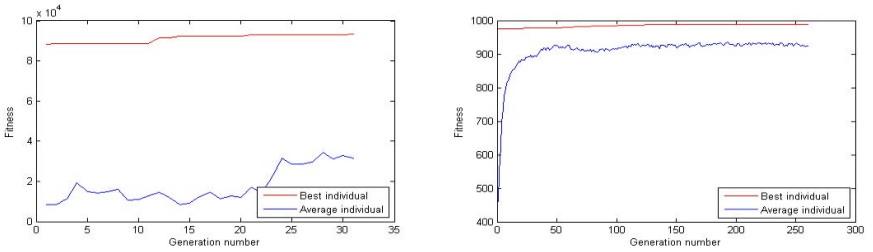
4.1 Duration Experiments

This section describes duration estimation experiments. The fitness was linearly dependent on the estimation error. The average estimation deviation of the best

individual from the training set cases was about ten percent of actual results. The duration values of the validation set were in the interval $\langle 0, 1994092411 \rangle$. Because of this enormous range, we did not use the absolute deviation between estimation and actual results for the validation error calculation. Instead, we used the logarithm difference of the estimation and actual results. Duration often vary significantly, even between similar tasks, and estimation need not to be too precise, therefore we consider this error calculation sufficient. The mean error on the validation set was 0.7, the standard deviation error was 0.9. Validation results of the best individual are shown in the Fig. 4.

4.2 Accuracy Experiments

The fitness was linearly dependent on the estimation error. The RMSE values of the validation set were in the interval $\langle 0, 1 \rangle$. The best individual's error on the training set was approximately 0.31. It is clear from the graph of the best run that there were lots of best individual improvements during computation, some of them were quite significant. The mean error on the validation set was 0,120234505 (which is near to the error on the training set) and the standard deviation was 0,18667105. Validation results of the best individual are shown in the Fig. 5.



(a) Best duration estimation run (b) Best accuracy estimation run

Fig. 3. Best GP runs

5 Conclusions

A GP algorithm capable of finding an estimation function of model performance was introduced in this paper. We have made the decision to split the performance estimation into the accuracy and duration estimation. The GP goal was to find two trees which estimate duration and accuracy performance of a model on the new data mining task. A metric for finding similar tasks was introduced. Terminals for obtaining model's previous performance results — including those utilizing metadata — were defined.

Experiments to find accuracy and estimation trees were performed. The results of these experiments suggest that genetic programming has the potential

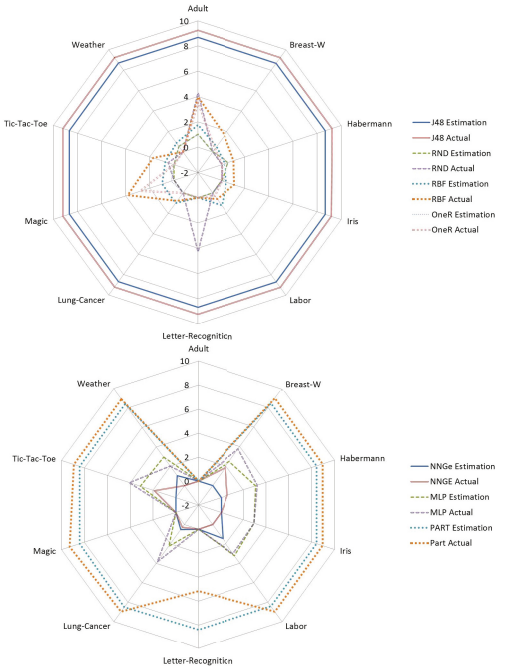


Fig. 4. Validation results of duration experiments

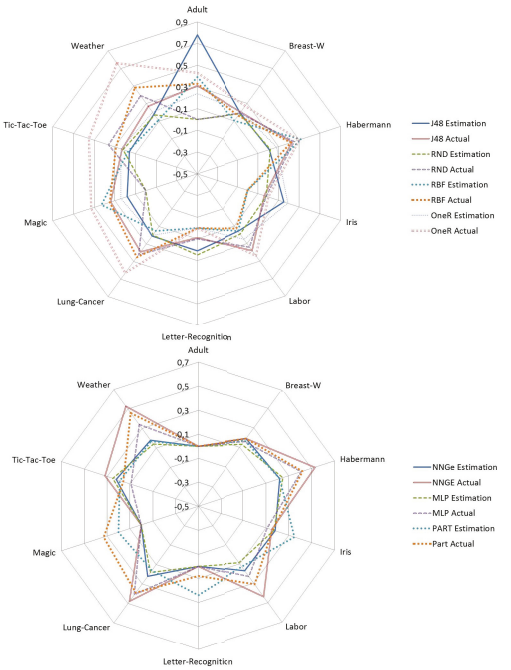


Fig. 5. Validation results of accuracy experiments

to be successfully used in metalearning. In our future work we intend to propose new metrics and analyze their time complexity. Testing on more datasets, including artificial ones is also crucial for solid validation of our technique. A comparison to other approaches, such as [5] is also planned.

Acknowledgment. Roman Neruda has been supported by the Czech Republic Science Foundation under the grant no. P202/11/1368, Jakub Šmíd has been supported by the project no. SVV2673/4.

References

1. Wolpert, D.H.: The supervised learning no-free-lunch theorems. In: Proc. 6th Online World Conference on Soft Computing in Industrial Applications, pp. 25–42 (2001)
2. Vilalta, R., Drissi, Y.: A perspective view and survey of meta-learning. Artificial Intelligence Review 18, 77–95 (2002)
3. Soares, C., Brazdil, P.B.: Zoomed Ranking: Selection of Classification Algorithms Based on Relevant Performance Information. In: Zighed, D.A., Komorowski, J., Żytkow, J.M. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 126–135. Springer, Heidelberg (2000)
4. Kazík, O., Pesková, K., Pilát, M., Neruda, R.: Meta learning in multi-agent systems for data mining. In: International Conference on Intelligent Agent Technology, pp. 433–434 (2011)
5. Graff, M., Poli, R.: Practical performance models of algorithms in evolutionary program induction and other domains. Artif. Intell. 174(15), 1254–1276 (2010)
6. Šmíd, J.: Agent optimization by means of genetic programming. Master's thesis, Charles University in Prague, Prague, Czech Republic (2012)
7. Frank, A., Asuncion, A.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2010), `http://archive.ics.uci.edu/ml`
8. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The weka data mining software: an update. SIGKDD Explor. Newsl. 11(1), 10–18 (2009)

Multi-caste Ant Colony Algorithm for the Dynamic Traveling Salesperson Problem

Leonor Melo[1,2], Francisco Pereira[1,2] and Ernesto Costa[2]

[1] Instituto Superior de Engenharia de Coimbra, 3030-199 Coimbra, Portugal
[2] Centro de Informática e Sistemas da Universidade de Coimbra, 3030-790 Coimbra, Portugal
leonor@isec.pt, {xico,ernesto}@dei.uc.pt

Abstract. In this paper we apply a multi-caste ant colony system to the dynamic traveling salesperson problem. Each caste inside the colony contains its own set of parameters, leading to the coexistence of different exploration behaviors. Two multi-caste variants are proposed and analyzed. Results obtained with different dynamic scenarios reveal that the adoption of a multi-caste architecture enhances the robustness of the algorithm. A detailed analysis of the outcomes suggests guidelines to select the best multi-caste variant, given the magnitude and severity of changes occurring in the dynamic environment.

Keywords: Ant Colony Optimization, Dynamic Traveling Salesperson Problem, Multi-caste Ant Colony System.

1 Introduction

Ant Colony Optimization (ACO) is one of the most successful branches of swarm intelligence. ACO algorithms were originally proposed by Marco Dorigo and, as its name suggests, take inspiration from pheromone-based interactions occurring in ant societies [4]. ACO algorithms mimic the natural foraging behavior, when solving a given problem. In concrete, artificial ants iteratively construct solutions biased by pheromone values and heuristic information. The pheromone values represent dynamic feedback information, reflecting the shared colony search experience and implementing a mechanism for indirect communication, whereas heuristic knowledge is usually modeled as problem specific greedy information.

Currently there are many ACO variants, with differences, e.g., in the strategy adopted by ants when building solutions or in the pheromone update rules [4]. Also, one must carefully define parameter settings when applying an ACO algorithm to a specific optimization situation as it is well-known that search performance is strongly affected by this choice. Selecting the ideal configuration for a given scenario is far from trivial and this difficulty is amplified when dealing with dynamic problems. In this case, the search environment changes over time and ants must quickly adapt to new situations. We hypothesize that an ACO algorithm with self-adaptive ability is better equipped to deal with changing environments. The multi-caste Ant Colony System (ACS), recently proposed

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 179–188, 2013.

in [14], is the adaptive algorithm considered in this paper. This framework divides the colony in several castes, each one with its own q_0 value, a parameter that strongly influences ACS search behavior. To test our hypothesis, we apply the multi-caste ACS to the Dynamic Traveling Salesperson Problem (DTSP), where the travel cost between pairs of cities changes over time. The analysis of the results helps to gain insight into the advantages of relying on a multi-caste framework to tackle dynamic environments.

The paper is structured as follows: in section 2 we present the multi-caste ACS used in our work, whereas in section 3 we introduce the dynamic TSP. Section 4 comprises the experimentation and analysis. Finally, section 5 gathers the main conclusions and suggests directions for future work.

2 Multi-caste Ant Colony System

Ant Colony System (ACS) was originally proposed in [3]. It differs from the original Ant System in a key issue: ants rely on a greedy decision rule to select the components that will appear in the solution. To balance the exploitation introduced by this rule, ACS contains a local pheromone update rule to discourage ants to follow the same trail (consult the aforementioned reference for details).

The application of ACS to a given problem requires the definition of the following parameters: m - Number of ants in the colony; β - relevance given to the heuristic knowledge; ρ - evaporation rate; q_0 - probability of selecting the next component greedily; ξ - pheromone decay coefficient; τ_0 - initial pheromone value. The parameter q_0 is essential in ACS, as it balances the relative importance given to exploration versus exploitation. Whenever an ant has to make a decision about which path to follow, a variable q is sampled: if $q < q_0$ the most promising edge is greedily chosen (exploitation); otherwise, an edge is probabilistically selected.

2.1 Our Approach

The multi-caste version of the ACS was originally proposed in [14]. In this work, it was applied to the static TSP and results showed that multi-caste architectures are more robust than standard ACS and effectively avoid the poor performance that follows as a consequence of a suboptimal selection of parameters.

In the multi-caste framework, ants belonging to the colony are divided in subgroups. The term caste is inspired by the behavior of biological ants and it was first used for artificial ants in [1]. Ants belonging to different castes have different q_0 values. The idea behind multi-caste ACS is to grant the algorithm different search strategies for different optimization periods. The framework comprises two variants: const-multi-caste and jump-multi-caste. The alterations needed to the conventional ACS are minimal.

Const-multi-caste: The colony is divided in castes, all with the same number of ants. The distribution remains fixed throughout the optimization. Each caste

is characterized by a specific q_0 value. When applying the state transition rules, each ant relies on the q_0 value from the caste to which it belongs.

Jump-multi-caste: The initial distribution is similar to that of const-multi-caste. However, at the end of each iteration, two ants are selected at random. If the ants belong to different castes and both castes have more than 20% of the total number of ants, the quality of their solutions is compared. The ant with the worse solution jumps to the caste of the winning ant. The aim is to provide a simple method to dynamically adjust the size of the castes, favoring those that in the current search status encode the most promising q_0 value.

Multi-caste ACS was originally proposed for the static TSP and a few alterations are needed to adjust it to a dynamic environment. In the scenarios addressed in section 4, the cost between pairs of cities changes over time. Whenever a change occurs we need to load a new distance matrix and recompute the nearest neighbor list. Also, the value of the best solution found needs to be recomputed to reflect the new costs. This is required since the same tour after the change could be associated with a larger travel distance. Keeping an old (untrue, but smaller) value would prevent the algorithm from updating the best-so-far ant, thus a sub-optimal solution would be used to update the trail.

3 Dynamic Traveling Salesperson Problem

The TSP is a famous NP-hard combinatorial optimization problem. Given a set of cities and all pairwise distances between them, the goal is to discover the shortest tour that visits every city exactly once. This was the first problem addressed by ACO algorithms, both because it is a difficult optimization situation and it can be modeled in a suitable way for the exploration performed by artificial ants (consult [4] for a detailed overview).

Two types of dynamism can added to the TSP: adding / removing cities to the problem or changing the cost between pairs of cities (i.e., inserting traffic jams). In this work we address the second modification, a variant known as DTSP with a traffic factor [12]. For each pair of cities, i and j, $e_{ij} = d_{ij} \times f_{ij}$, where d_{ij} is the original distance between cities i and j, and f_{ij} is the traffic factor between those cities. Every F evaluations, a random number R in $[F_L, F_U]$ is generated probabilistically. F_L and F_U are the lower and upper bounds for the traffic factor and R represents the traffic at that moment. With a probability M each link can change its traffic factor to $f_{ij} = 1 + R$, or otherwise, reset its traffic factor to 1 (meaning no traffic). F and M represent the number of $\frac{evaluations}{10}$ between changes (i.e., its frequency) and the magnitude of change, respectively. For all experiments reported in this paper, we consider 16 dynamic scenarios (4 values of $F\times$ 4 values of M): $F = \{10, 20, 100, 200\}$, where $F = 10$ defines a rapid changing environment and $F = 200$ represents a slow changing environment; $M = \{10, 25, 50, 75\}$, with $M = 10$ and $M = 75$ establishing a small and large degree of change, respectively.

To compare the different algorithms, we adopt the offline performance [2], which consists on the average performance of the best-since-last-change solution at each time step, as described in formula 1:

$$P_{offline} = \frac{1}{E} \sum_{i=1}^{E} \left(\frac{1}{T} \sum_{j=1}^{T} P_{ij}^{*} \right) \tag{1}$$

where E is number of iterations in each run, T is the number of independent runs, and P_{ij}^{*} is the best solution at iteration i of run j considering only the ants that existed since the last change of the present run.

3.1 ACS for the Dynamic Traveling Salesperson Problem

ACO variants can benefit from on-line parameter adjusting even in static TSP instances [17], [14]. We expect that this advantage is more visible in dynamic situations. To confirm this supposition, we selected a TSP instance with 200 cities and compared the offline performance of several mono-caste configurations $q_0 = \{0.75, 0.9, 0.95, 0.99\}$, in different frequency and magnitude of change scenarios. In figure 1 we present an overview of the analysis. Results are the averages of 30 runs and each run went through 30 variations. For each one of the 16 scenarios, we compared every pair of configurations using the paired t-test with confidence level of 0.95 and 29 degrees of freedom. The value in any given cell represents the difference between the number of configurations that were found to be significantly worse, and the number of configurations significantly better than that cell. The lighter the color (and higher the number), the better the configurations performed when compared to the others.

Some values of q_0 seem to be better suited for a given scenario than others. For instance, $q_0 = 0.99$ is the best when the the magnitude of change is high (each link with a 50% to 75% chance of change) and the change occurs frequently (every 10 or 20 iterations), but behaves poorly when both the magnitude and frequency of change are low. On the other hand, $q_0 = 0.95$ seems to have a more balanced behavior but it favors the low frequency/high magnitude and high frequency/low magnitude environments, while $q_0 = 0.90$ is at its best in low frequency/low magnitude scenarios. Similar results were obtained with other instances. This outcome suggests that the coexistence of different q_0 is beneficial, as it provides the algorithm with tools to change its search behavior, thereby adapting to dynamic environments.

3.2 Related Work

The existing ACO approaches for the DTSP cover both forms of dynamism: addition/elimination of cities [6], [8], [7], [16], [10], [11], as well as the modification of the cost between pairs of cities (inserting traffic jams): [5], [9], [13], [12]. Several dynamic features have been addressed by ACO approaches. A non-exhaustive list of variations includes:

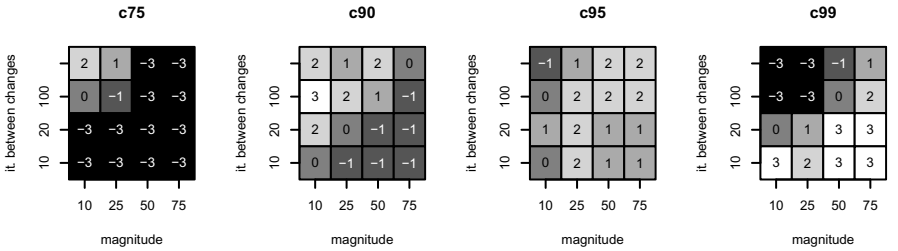


Fig. 1. ACS mono-caste performance for a dynamic TSP instance of size 200

- frequency change - from short to long intervals between changes (from 20 iterations [10], [13], [12], [11] to 750 iterations [8], [7] or somewhere in between [16]); also from a single to several simultaneous changes [6], [5], [9];
- severity of change - from 0.5% [8], [7] to 75% [10], [11] of the cities added/ removed; from 1% [5], [9] to 75% [13], [12] of the links affected by traffic jams;
- cycle length and accuracy [13];

Although there is sometimes a changing pattern (predictability of change), no algorithm explores it. Also, the dimension of the instances used in tests is highly variable, ranging from situations with 25 cities [9] to those with 532 [13]. There is not a widely accepted DTSP benchmark in the literature making it impossible to directly compare the algorithm.

There are different strategies adopted by ACO approaches to deal with change. These include: trail equalization or adjustment [6], [8], [16], [5]; a local search procedure (KeepElitist) applied to an ant or a group of ants when a change occurs [8], [7], [16], [10], [11], [12], [13]; the existence of explicit memory [7], [16], [12], [13]; the implementation of an immigrants scheme [10], [11], [12], [13]. Finally, several ACO variants have been adopted for dynamic situations. Some noteworthy examples include Ant System [6], [8], [5], Ant Colony System [9], [16] or P-ACO [7], [10], [11], [12], [13].

4 Experiments

We selected the TSP instances kroA200, and att532 from the TSPLIB 95 [15] to build the 16 dynamic scenarios as described in section 3. Unless otherwise noted, the default values used for the experiments are: $m = 10$, $\beta = 2$, $\rho = 0.1$, $\xi = 0.1$, $\tau_0 = 1/(n \cdot L_{nn})$ (where L_{nn} is the length of the tour using the nearest neighbor heuristic [4]), $q_0 = \{0.75, 0.9, 0.95, 0.99\}$, and the local search algorithm is the 3-opt. In multi-caste configurations, all castes have the same (initial) size. Each experiment was repeated 30 times and was allowed to run for at least $30 \times F \times 10$ evaluations (minimal adjustments to this number were made to allow a fair

comparison with the configurations containing 8 and 12 ants). Configurations are identified according to the following convention:

- cx: mono caste configuration with a q_0 value of 0.x (eg.: c95);
- cx_y (jx_y): constant (jump) dual-caste configuration with q_0 values of 0.x and 0.y (eg.: c90_95);
- c$x$quads$m$ (j$x$quads$m$): constant (jump) quad-caste configuration with q_0 values of 0.x, 0.90, 0.95, and 0.99 and a total of m ants (eg.: c75quads12);

For all scenarios, we compared each pair of configurations using the paired t-test with confidence level of 0.95 and 29 degrees of freedom. In the upcoming figures, values in cells represent the difference between the number of configurations that discovered an average tour length larger (worse) than the current setting and the number of configurations with a smaller average tour length. Higher values identify better configurations. Shaded cells highlight combinations whose behavior is not statistically different from the one that achieved the best result in that particular environment (confidence level: 0.05).

Figure 2 contains the comparative study concerning the results obtained in the kroA200 instance. The configurations with best performance (considered as those that are statistically equivalent to the one with the best offline performance) are c95_99, c95, c99, j95_99, c90, and j90_99. The worst configurations are c50_99, c75_99, and c50quads, suggesting that a too low q_0 value compromises the reaction of the algorithm in dynamic situations. Results clearly show that the behavior of the mono-caste configurations tends to be more extreme. Even thought there are several scenarios where their performance is close to the best, c90 and c99 also have some of the worst results. This outcome confirms that standard ACS requires a careful definition of parameters, in order not to compromise its performance. On the contrary, several multi-caste configurations (e.g., c95_99, c90_99, and to some extent j75quads) are better equipped to prevent bad results. The jump-caste mechanism seems to be important in the j50_99, j75_99 and jquads configurations, as it allows the migration of ants to the caste with $q_0 = 0.99$, if the environment requires it. This effect is most visible in scenarios with a very high frequency and magnitude of change.

The comparative study concerning the results obtained in the att532 instance is displayed in figure 3. This is a larger TSP instance with 532 cities and a brief perusal of the figure reveals that a configuration (c99) is better than all others. On the contrary, c90 and most const multi-caste configurations have the worst performance. This is another example of a situation where ACS is very sensitive to the definition of an appropriate q_0 value. As such, mono-caste configurations with the wrong choice of q_0 lead to poor result. Multi-caste ACS variants, by allowing the coexistence of several q_0 are better equipped to deal with this situation. In most of the scenarios from this instance, the jump-caste variant outperforms its constant counterpart This situation is probably related to the advantage in defining aggressive q_0 for this particular situation. An inspection of figure 3 shows that configurations j95_99, j90_99, c95_99, and jquads12 are able to avoid bad results, while maintaining a reasonable quality.

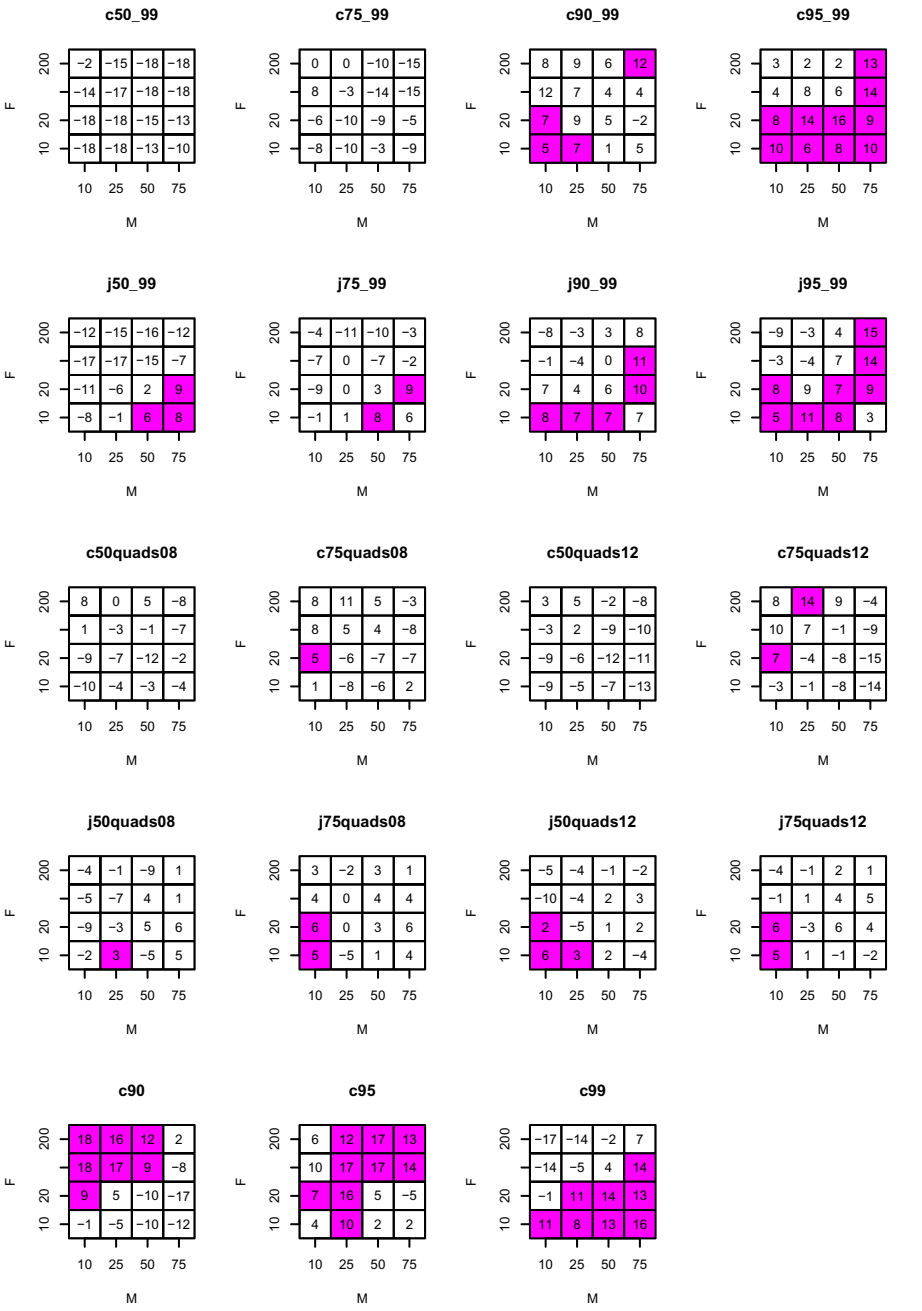


Fig. 2. Comparative performance of the ACO configurations in the 16 optimization scenarios from the kroA200 instance

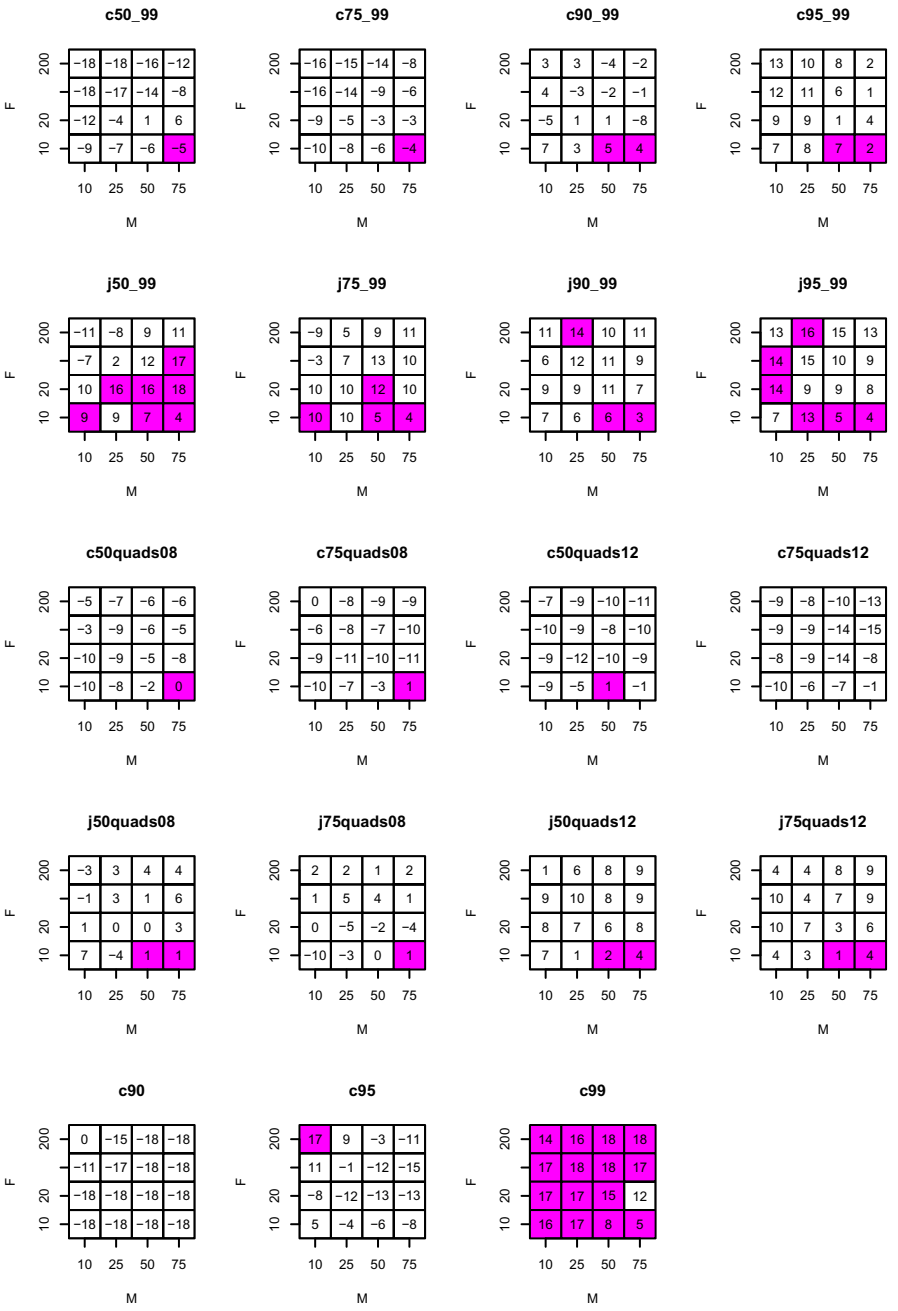


Fig. 3. Comparative performance of the ACO configurations in the 16 optimization scenarios from the att532 instance

To complete our analysis, we measured the peaks in the length of the tour that appear immediately after a change, to determine which configurations react faster to the change. Equation 2 is used in this study:

$$Q = \frac{1}{E} \sum_{i \in C} \left(\frac{1}{T} \sum_{j=1}^{T} P_{ij} \right) \tag{2}$$

where C is the set of iterations immediately after a change, T is the number of independent runs, and P_{ij} is the best solution found at iteration i of run j.

We then compared, for every scenario, every pair of configurations according to the average peak value. In this case the differences between the distributions is quite marked. In all TSP instances, the configurations from the quads12 group have the lowest peaks after a change. The configurations from the quads08 have, consistently, the largest peaks. The differences between quads12, quads08, and the rest of the configurations are evident: for almost all scenarios, every element of the quads12 is similar to the best, every element in quads08 is similar to the worst and the remaining configurations are different from both the best and the worst.

5 Conclusions

In this paper we studied the application of a multi-caste ACS to the dynamic TSP. This framework divides the ants in different castes, each one with its own q_0 value, thereby promoting the appearance of different search behaviors.

Results show that, as a rule, the existence of several castes enhances the robustness of the algorithm when solving dynamic situations. Even though multi-caste configurations are usually not the absolute best algorithms, the coexistence of simultaneous q_0 values prevents them from very poor performances. Also, the existence of different castes removes the need to carefully define q_0, which is an essential parameter for the success of standard ACS. Overall, the c95_99 configuration seems to provide a compromise, as it always avoids poor performances, independently of the problem instance and of the dynamic scenario properties. The j95_99 and j75quads12 also showed a robust behavior while maintaining reasonably good results. Additionally, the quads12 castes are effective in avoiding big degradation in performance immediately after a change.

In the near future we aim to apply the existing multi-caste framework to other dynamic situations to gain additional insight into its main strengths and weaknesses. Also, we will investigate how the algorithm performs in cyclical dynamic environments. In what concerns the multi-castes modus operandi, the current jump-mechanism seems to be too slow in re-balancing the caste sizes. This can be easily explained since, the larger the dominant caste is, the harder it will be for an ant from the smaller cast (even if producing a better solution), to be randomly selected. Testing alternative rules for the immigration of ants between castes is another topic that we will address in our research.

References

1. Botee, H., Bonabeau, E.: Evolving ant colony optimization. Advances in Complex Systems (1998)
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers (2002)
3. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
4. Dorigo, M., Stützle, T.: Ant Colony Optimization. A Bradford Book, MIT Press, Cambridge, Massachussetts (2004)
5. Eyckelhof, C.J., Snoek, M.: Ant systems for a dynamic TSP: Ants caught in a traffic jam. In: ANTS 2002: Third International Workshop (2002)
6. Guntsch, M., Middendorf, M.: Pheromone Modification Strategies for Ant Algorithms Applied to Dynamic TSP. In: Boers, E.J.W., Gottlieb, J., Lanzi, P.L., Smith, R.E., Cagnoni, S., Hart, E., Raidl, G.R., Tijink, H. (eds.) EvoIASP 2001, EvoWorkshops 2001, EvoFlight 2001, EvoSTIM 2001, EvoCOP 2001, and EvoLearn 2001. LNCS, vol. 2037, pp. 213–222. Springer, Heidelberg (2001)
7. Guntsch, M., Middendorf, M.: Applying Population Based ACO to Dynamic Optimization Problems. In: ANTS 2002: Third International Workshop, pp. 111–122 (2002)
8. Guntsch, M., Middendorf, M., Schmeck, H.: An ant colony optimization approach to dynamic TSP. In: GECCO 2001 Proceedings of the Genetic and Evolutionary Computation Conference, pp. 860–867. Morgan Kaufmann Publishers (2001)
9. Liu, J.L.: Rank-based ant colony optimization applied to dynamic traveling salesman problems. Engineering Optimization 37(8), 831–847 (2005)
10. Mavrovouniotis, M., Yang, S.: Ant Colony Optimization with Immigrants Schemes in Dynamic Environments. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN XI. LNCS, vol. 6239, pp. 371–380. Springer, Heidelberg (2010)
11. Mavrovouniotis, M., Yang, S.: A memetic ant colony optimization algorithm for the dynamic travelling salesman problem. Soft Computing 15(7), 1405–1425 (2011)
12. Mavrovouniotis, M., Yang, S.: An Immigrants Scheme Based on Environmental Information for Ant Colony Optimization for the Dynamic Travelling Salesman Problem. In: Hao, J.-K., Legrand, P., Collet, P., Monmarché, N., Lutton, E., Schoenauer, M. (eds.) EA 2011. LNCS, vol. 7401, pp. 1–12. Springer, Heidelberg (2012)
13. Mavrovouniotis, M., Yang, S.: Memory-Based Immigrants for Ant Colony Optimization in Changing Environments. In: Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A.I., Merelo, J.J., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N. (eds.) EvoApplications 2011, Part I. LNCS, vol. 6624, pp. 324–333. Springer, Heidelberg (2011)
14. Melo, L., Pereira, F., Costa, E.: Multi-caste Ant Colony Optimization Algorithms. In: Proceedings of the 15th Portuguese Conference on Artificial Intelligence, Lisbon, pp. 978–989 (2011), `http://epia2011.appia.pt/LinkClick.aspx?fileticket=s--Tr74EzUI%3d&tabid=562`
15. Reinhelt, G.: {TSPLIB}: a library of sample instances for the TSP (and related problems) from various sources and of various types, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`
16. Sammoud, O., Solnon, C., Ghedira, K.: A New ACO Approach for Solving Dynamic Problems. In: 9th International Conference on Artificial Evolution (2009), `http://liris.cnrs.fr/publis/?id=4330`
17. Stützle, T., Lopez-Ibanez, M., Pellegrini, P., Maur, M., de Oca, M.M., Birattari, M., Dorigo, M.: Parameter Adaptation in Ant Colony Optimization. Technical report number tr/iridia/2010-002, IRIDIA, Bruxelles, Belgium (2010)

Generalized Information-Theoretic Measures for Feature Selection

Davor Sluga and Uros Lotric

University of Ljubljana, Faculty of Computer and Information Science,
Trzaska 25, 1000 Ljubljana, Slovenia
{davor.sluga,uros.lotric}@fri.uni-lj.si

Abstract. Information-theoretic measures are frequently employed to select the most relevant subset of features from datasets. This paper focuses on the analysis of continuous-valued features. We compare the common approach with discretization of features prior the analysis, to the direct usage of exact values. Due to the overwhelming costs of computing continuous information-theoretic measures based on Shannon entropy the Renyi and Tsallis generalized measures are considered. To enable computation with continuous Tsallis measures a novel modification of the information potential is introduced. The quality of the analysed measures was assessed indirectly through the classification accuracy in conjuction with the greedy feature selection process. The experiments on datasets from UCI repository show considerable improvements of the results when using both generalized continuous measures.

Keywords: Feature selection, Information theory, Renyi entropy, Tsallis entropy.

1 Introduction

In the field of data mining and knowledge discovery we are usually involved in analysis of data attributes or features. In many cases we use supervised learning techniques to build classification models which associate instances to classes. Despite the fact that all available features describe instances in maximum available detail, many of them are partially or completely irrelevant for the association. Besides, a huge number of features usually leads to very long processing times and models with poor generalization capabilities [1]. By using feature selection methods we try to obtain only the relevant features and consequently improve the efficiency and generalization capabilities of the classification models. Numerous feature selection methods exist [2], some are fast and wide-ranging, but lack in effectiveness, others give better results but are computationally expensive. A majority of them need discrete data to perform feature selection. If data includes continuous features, they must be discretized beforehand, which causes varying results, depending on the discretization method used. There are many types of evaluation functions used in the feature selection process; a review can be found in [1].

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 189–197, 2013.

Information measures are definitely among the most interesting because they arise from the definition of feature selection: we wish to obtain a set of features providing the largest amount of information on the problem domain. Commonly used measures include average mutual information [3] and its extensions [4,5]. Measures based on generalizations of Shannon entropy, are also being considered [6].

Due to the computational considerations mainly the discrete version of the entropies are used in practice [7]. This poses aforementioned problem when analysed features are continuous in their nature. Our idea is to find a proper information measures and algorithms to support direct analysis of continuous-valued features. We propose using two evaluation functions based on the Renyi [8] and Tsallis entropy [9] that can be used to evaluate feature relevancy and can cope with continuous as well as discrete data.

In the next section the information-theoretic measures are presented together with proposed modifications, which make them ready for feature selection. Section three presents the greedy algorithm used to sweep through feature space. In section four the performance of different information-theoretic feature selection methods are indirectly (through a classifier) evaluated on some datasets from UCI repository. Main findings are summarized at the end.

2 Information-Theoretic Measures

Supervised learning is a common task in machine learning, where one tries to build a model which relates an instance to a class. Information theory measures offer means to rank features according to the information they provide about class values [5].

For a set of features $\mathbf{X}$, which can acquire values $\mathbf{x}_1, ..., \mathbf{x}_n$ with probabilities $p(\mathbf{x}_1), ..., p(\mathbf{x}_n)$, we can calculate the Shannon entropy as

$$H_S(\mathbf{X}) = -\sum_{i=1}^{n} p(\mathbf{x}_i) \log_2 p(\mathbf{x}_i) \ . \tag{1}$$

In a similar manner we can calculate the entropy of class $H_S(C)$ given the possible class values $C = \{c_1, \ldots, c_m\}$ with probabilities $p(c_1), \ldots, p(c_m)$ as well as the joint entropy $H(\mathbf{X}, C)$ given the joint probabilities $p(\mathbf{x}_i, c_j)$ of relating the instance $\mathbf{x}_i$ to the class value c_j. Provided with the above values we can use mutual information

$$I(\mathbf{X}; C) = H_S(\mathbf{X}) + H_S(C) - H_S(\mathbf{X}, C) \tag{2}$$

to assess the relevancy of a set of features $\mathbf{X}$ with regard to the class C.

Usually the probability distributions are not known and need to be estimated from instances in the data. If the features are continuous in their nature, a discretization step is usually applied to the data beforehand. Unfortunately the discretization can lead to spurious results by shrouding some proprieties of the probability distribution [10].

To avoid the discretization step we can use the differential entropy

$$H_S^D(\mathbf{X}) = -\int p(\mathbf{x}) \log_2 p(\mathbf{x}) d\mathbf{x} \tag{3}$$

with $p(\mathbf{x})$ being the probability density function. The numerical computation of differential entropy for a multi-dimensional set of features is computationally expensive; in three or more dimensions the usage of Monte Carlo integration is almost a must.

The non-parametric Parzen window method [8] is the most straightforward approach for estimating the probability density function from the data. The estimate is obtained by spanning kernel functions around the instances, $p(\mathbf{x}) = \sum_{i=1}^{n} G(\mathbf{x}-\mathbf{x}_i, \mathbf{h})/n$. If we assume a multivariate normal distribution of the data, we can use a product of gaussians $G(\mathbf{x}, \mathbf{h}) = \prod_d G(x_d, h_d)$ as the kernel function. According to the Silverman's rule the width of a gaussian $h_d = 1.06\sigma_d n^{-1/5}$ in each dimension d of the feature vector depends on the distribution of the instance values given in terms of standard deviation σ_d .

Besides the classical Shannon entropy there exists a range of entropy generalizations. Two of the more widely known are the Renyi entropy [8] and Tsallis entropy [9]

$$H_{R_q}(\mathbf{X}) = \frac{1}{1-q} \log \sum_{i=1}^{n} p(\mathbf{x}_i)^q \quad , \quad H_{T_q}(\mathbf{X}) = \frac{1}{q-1} \left(1 - \sum_{i=1}^{n} p(\mathbf{x}_i)^q \right) \ , \tag{4}$$

which extend the original concept by introducing an additional parameter q. It should be noted that both entropies converge to Shannon entropy as q approaches 1 in the limit. They can replace the original Shannon entropy in calculation of the average mutual information [11]. However, when substitutions are used we can no longer speak of mutual information, but rather mutual entropy since the presented entropies have not been theoretically established as measures of information [9].

Similarly as in the case of Shannon entropy, the differential versions of the two generalized entropies are also defined, with the sum $\sum_{i=1}^{n} p(\mathbf{x}_i)^q$ substituted by the integral $\int p(\mathbf{x})^q d\mathbf{x}$ in Eq. 4. Hild et al. [8] showed that by setting $q = 2$ and approximating the probability density function using Parzen windows with gaussian kernels, we can simplify the computation. Namely, the information potential $V(\mathbf{x}) = \int p(\mathbf{x})^2 d\mathbf{x}$ can be estimated as

$$\hat{V}(\mathbf{x}) = \frac{1}{n^2} \sum_{k=1}^{n} \sum_{j=1}^{n} G(\mathbf{x}_k - \mathbf{x}_j, \sqrt{2}\mathbf{h}) \ . \tag{5}$$

The summation heavily reduces the computational time compared to the Monte Carlo numerical integration of Shannon entropy. The Renyi differential quadratic entropy estimator thus becomes

$$\hat{H}_{R_2}^D(\mathbf{X}) = -\log \hat{V}(\mathbf{x}) \ . \tag{6}$$

The same strategy can not be used directly in the case of Tsallis differential entropy. Namely, the sum in Tsallis entropy is always smaller or equal to 1 for $q > 1$, which ensures that the entropy is always non-negative. On contrary, there is no such guarantee for the Tsallis differential entropy. From the properties of the product of gaussian functions and Eq. 5 we can see that the upper limit for the estimated information potential $\hat{V}(\mathbf{x})$ equals $\prod_d (2\sqrt{\pi}h_d)^{-1}$. For some choices of widths h_d, the upper limit can easily exceed 1, which can potentially lead to negative values of Tsallis differential entropy. This makes the mutual comparison of two feature vectors of different sizes impossible. Namely, analysis of an additional dimension in the feature vector can reverse the sign of differential Tsallis entropy, pointing to an illogical conclusion that with an additional feature less is known about the problem.

To overcome this issue we propose a new normalized estimator for the information potential

$$\hat{V}_T(\mathbf{x}) = \left(2^d \pi^{d/2} \prod_d h_d\right) \hat{V}(\mathbf{x}) \ , \tag{7}$$

which is confined to the interval $[0, 1]$. As a consequence, the proposed estimator for the normalized Tsallis differential quadratic entropy equals

$$\hat{H}^D_{T_2}(\mathbf{X}) = 1 - \hat{V}_T(\mathbf{x}) \tag{8}$$

and is normalized to the same interval. The normalized information potential estimator could be also used in computation of Renyi differential entropy, where the normalization factor produces only a constant offset.

3 Greedy Feature Selection

Feature selection methods are presented with a full set of possible candidate features and must choose an optimal subset according to some evaluation function. When searching through a set of n candidate features, the feature selection algorithm should evaluate 2^n possible feature subsets in order to find the optimal one. This search strategy is practically prohibitive, even for a few dozen candidate features. Consequently, other types of search strategies are used, which drastically reduce the processing time at the cost of the possibility of selecting suboptimal feature subset. Examples of such strategies are genetic algorithms [12], greedy algorithms [13], simulated annealing [14] and branch and bound algorithms [15]. In our paper we will focus on the greedy algorithms, specifically on the Sequential Forward Selection algorithm [13], which presents one of the simplest search strategies for feature selection.

The Sequential Forward Selection algorithm (Fig. 1) starts with an empty set of features and at each step sequentially adds a feature x that maximizes the measure of relevancy $I(\mathbf{X} \cup x; C)$ when combined with a set features $\mathbf{X}$ that have already been selected.

```
begin
  start with the empty set X = ∅
  repeat
    set X_0 = X
    select the next best feature x_best = argmax_{x∉X_0} I(X_0 ∪ x, C)
    set X = X_0 ∪ x_best
  until I(X, C) ≤ I(X_0, C)
end
```

Fig. 1. Sequential Forward Selection algorithm

Sequential Forward Selection algorithm is a simple search strategy and works best when the optimal subset of features is small. It's main disadvantage is that it is unable to remove features that become obsolete after the addition of other features. It can be improved by the similar Sequential Backward Selection and the Sequential Floating Selection algorithms [16].

4 Experimental Work

We compared the feature selection performance for discrete and differential entropies of Shannon, Renyi and Tsallis. We employed them in terms of average mutual information or corresponding Renyi and Tsallis average mutual entropy as evaluation functions in the Sequential Forward Selection algorithm. From the top five features returned by the Sequential Forward Selection algorithm we built a classification tree using the Weka data mining tool implementation of C4.5 algorithm [17]. The classification accuracy of the model, given as a proportion of correctly classified instances, represented a measure of the quality of the selected features.

We tested the feature selection methods on three datasets which include discrete and continuous features and have a discrete class. Table 1 gives some details about Sonar, Ionosphere and Wave datasets from UCI Machine Learning repository [18].

Table 1. Properties of the selected datasets

Dataset	Number of features		Number of instances by class value		
	continuous	discrete	0	1	2
Sonar	60	0	111	97	0
Ionosphere	32	2	225	125	0
Wave	40	0	100	100	100

We had to discretize the continuous-valued features prior to using them with discrete entropy measures. Feature values of all instances were allotted to corresponding intervals; the non-interleaved intervals of equal size were covering the whole range of each features' values.

In cases when instances are described with discrete- and continuous-valued data, the differential entropies can be directly used on both. The time complexity of calculating the differential entropies is $O(n^2)$ in comparison to $O(n)$ of the discrete entropies. Consequently, when data is only discrete in its nature it is better to put differential entropies aside and only use entropies to reduce the computational costs.

During the feature selection we varied the number of instances, from 20 to maximum available, to see what effect it has on the selection of features and consequently on the classification accuracy. When evaluating the classification accuracy we randomly split all instances in a dataset into two parts, using 67% of instances for training and 33% of instances for testing. Each experiment was executed 100 times to obtain relevant results; in the following the average values are presented.

First we varied the number of discrete intervals from 3 to 9. Fig. 2a shows the comparison of using the discrete versions of the three types of entropies. We can see that using different number of intervals to discretize the data does not affect the feature selection process considerably. In most cases there is a slight improvement achieved by increasing the number of intervals since using a larger number of intervals reveals more variability of the data. However, by increasing the number of intervals and keeping the number of instances fixed, the assessment of probability density function in higher dimensions becomes inaccurate. This is why in some of the test cases discretization to nine intervals preforms worse than others. As expected, using more instances to perform feature selection improves the relevancy of the selected features, leading to better classification accuracy.

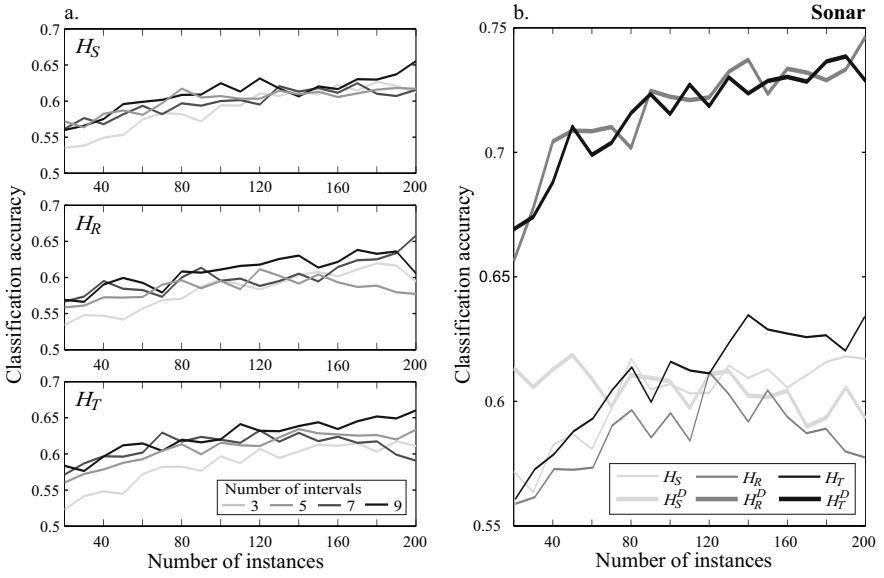


Fig. 2. Classification accuracy on the Sonar dataset

Fig. 2b shows that Renyi and Tsallis differential entropies perform much better than their discrete counterparts. For clarity only the classification accuracy obtained with discretization to five intervals is shown. Shannon differential entropy is on pair with the discrete entropies, except for low numbers of instances where it performs somewhat better. The main reason for poor performance of the Shannon differential entropy lies in the inaccurate but yet time consuming Monte Carlo integration. Interestingly, the Renyi entropy performance is the worst, contrasting with the Tsallis entropy, which performs even better than the Shannon entropy.

Fig. 3 shows similar results for the Ionosphere and Wave dataset. Renyi and Tsallis differential entropies perform quite well, achieving more than 10% higher classification accuracy than the rest. Shannon differential entropy does not improve with increasing number of instances. This might be attributed to the increasing complexity of the estimated probability density function, requiring a more exact but impractical integration process. Obviously using the Monte Carlo integration to compute the Shannon entropy is not as efficient as the computation of the two generalized differential entropies, since the two perform much better. Again Renyi entropy performed the worst on both datasets and Tsallis entropy achieved on average the best performance on the discretized datasets.

In all three datasets the differential entropies behave very well even for really small numbers of instances, which is not the case for the entropies.

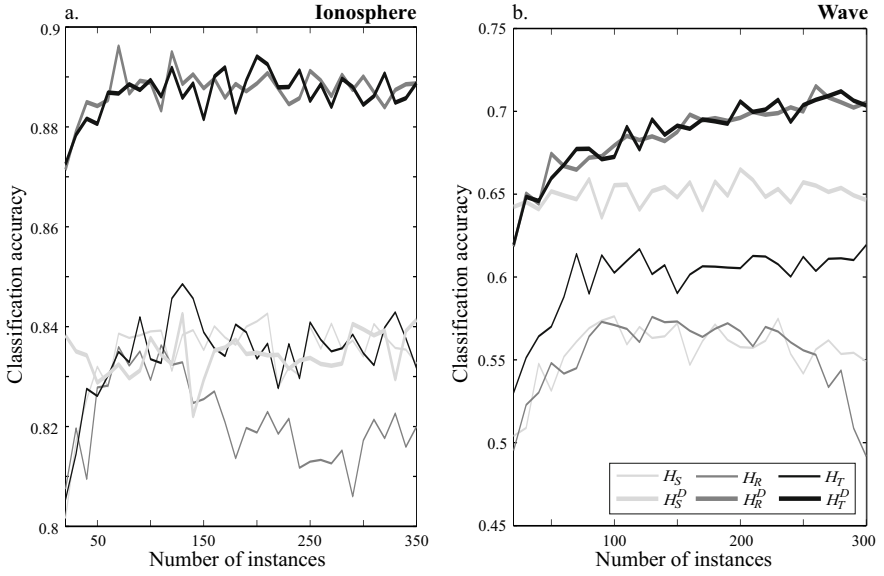


Fig. 3. Classification accuracy on Ionosphere and Wave dataset

5 Conclusion

In this paper we considered a problem of selecting features that most relevantly describe instances and relate them to a given class. We focused on the continuous-valued features and assessment of their relevancy. Opposed to the classical approach with discretization step we explored the possibility of directly determining features relevancy.

The experiments show that generalized entropies and generalized differential entropies considerably improve the relevancy of continuous-valued features in feature selection task — the classification accuracy is greatly improved in comparison with the classical Shannon approach.

The proposed normalization of the information potential used in Tsallis differential entropy makes it appropriate for the feature selection task. This puts it on pair with the Renyi differential entropy, as both achieve similar results in terms of classification accuracy. Besides, Tsallis average mutual entropy behaves much better than the Renyi mutual entropy on the discretized data. This confirms that Tsallis entropy and Tsallis diferential entropy are good choices for assessing the feature relevancy when dealing with discrete- as well as continuous-valued data, respectively.

More complex discretization strategies might better grasp the variability of data among instances and thus improve the relevancy of the selected features. Similarly, a lot of possibilities for improvements is also on the side of differential entropies, mainly in better approximation of probability density function and reduction of the computational costs. In our work we used a simple greedy algorithm as a search strategy, but more advanced exist, leaving even more room for further improvement of our methods.

References

1. Dash, M., Liu, H.: Feature Selection for Classification. Intelligent Data Analysis 1(1-4), 131–156 (1997)
2. Blum, A., Langley, P.: Selection of relevant features and examples in machine Learning. Artificial Intelligence 97(1-2), 245–271 (1997)
3. Lewis, D.: Feature selection and feature extraction for text categorization. In: Proceedings of Speech and Natural Language Workshop, pp. 212–217. Morgan Kaufmann, San Francisco (1992)
4. Liu, H., Sun, J., Liu, L., Zhang, H.: Feature selection with dynamic mutual information. Pattern Recognition 42(7), 1330–1339 (2009)
5. Estevez, P.A., Tesmer, M., Perez, C.A., Zurada, J.M.: Normalized Mutual Information Feature Selection. IEEE Transactions on Neural Networks 20(2), 189–201 (2009)
6. Lopes, F.M., Martins, D.C., Cesar, R.M.: Feature selection enviroment for genomic applications. BMC Bioinformatics 9(1), 451–458 (2008)
7. Fleuret, F.: Fast binary feature selection with conditional mutual information. Journal of Machine Learning Research (5), 1531–1555 (2004)
8. Hild Il, K.E., Erdogmus, D., Torkkola, K., Principe, J.C.: Feature Extraction Using Information-Theoretic Learning. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(9), 1385–1392 (2006)

9. Furuichi, S.: Information theoretical properties of Tsallis entropies. J. of Mathematical Physics 47(2) (2006)
10. Mejía-Lavalle, M., Morales, E.F., Arroyo, G.: Two Simple and Effective Feature Selection Methods for Continuous Attributes with Discrete Multi-class. In: Gelbukh, A., Kuri Morales, Á.F. (eds.) MICAI 2007. LNCS (LNAI), vol. 4827, pp. 452–461. Springer, Heidelberg (2007)
11. Vila, M., Bardera, A., Feixas, M., Sbert, M.: Tsallis Mutual Information for Document Classification. Entropy (13), 1694–1707 (2011)
12. Oh, O.I.-S., Lee, J.-S., Moon, B.-R.: Hybrid genetic algorithms for feature selection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(11), 1424–1437 (2004)
13. Jianping, H., Waibhav, T.D., Dougherty, E.R.: Performance of feature-selection methods in the classification of high-dimension data. Pattern Recognition 42(3), 409–442 (2009)
14. Lin, S.-W., Tseng, T.-Y., Chou, S.-Y., Chen, S.-C.: A simulated-annealing-based approach for simultaneous parameter optimization and featureselection of backpropagation networks. Expert Systems with Application 34(2) (2008)
15. Somol, P., Pudil, P., Kittler, J.: Fast branch & bound algorithms for optimal feature selection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(7), 900–912 (2004)
16. Tang, E.-K., Suganthan, P., Yao, X.: Gene selection algorithms for microarray data based on least square support vector machine. BMC Bioinformatics 7, 95 (2006)
17. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11(1) (2009)
18. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, School of Information and Computer Science, Irvine, CA (2012), `http://archive.ics.uci.edu/ml`

PCA Based Oblique Decision Rules Generating

Marcin Michalak[1,2] and Karolina Nurzyńska[1]

[1] Institute of Informatics, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland
{Marcin.Michalak, Karolina.Nurzynska}@polsl.pl
[2] SOMAR S.A., ul. Karoliny 4, 40-186 Katowice, Poland
M.Michalak@somar.com.pl

Abstract. The paper presents the new algorithm of oblique rules induction. On the basis of the initial step that consists in clustering the decision class into subclasses, for every subclass the oblique hypercuboid is generated. Sides of the hypercuboid are parallel and perpendicular to the directions defined by *PCA*. One hypercuboid corresponds to one decision rule. Results of inducting rules in the new way were compared with other oblique and non-oblique rules sets built on the artificial and real data.

Keywords: machine learning, rules induction, decision systems, Principal Component Analysis, oblique rules.

1 Introduction

Classification is one of the most popular problems in data analysis. It is expected, that it is possible to extract the dependence (or dependencies) between object features and the class that it belongs to. If we denote x as the object from the set X and C as the finite set of class labels l the classification task can be described as the function $f : X \to C$.

The problem of classifying objects can be solved in two ways. Sometimes it is not necessary to know how the classifier works and only the classification results are interesting. Algorithms that represent this approach can be described as „black boxes": one knows the result but one does not know the mechanism of reasoning. However, sometimes it is useful to know what kind of dependencies in the data generate the specific expression that assigns the given object to one of the class labels.

Each of the mentioned approaches is represented by a wide group of algorithms. The most popular „black boxes" are: support vector machines, artificial neural networks, k–nearest neighbor. More interpretable results come from decision trees or decision rules.

Here, a new way of oblique rules induction is presented. This kind of rules still gives us the possibility of understanding (or interpreting) of obtained model. However, it should be considered as the way of generalisation of typical (hypercuboidal) rules. The main motivation to develop this method is to decrease the

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 198–207, 2013.

rule description of dependencies in the data, assuring still the acceptable level of classification accuracy. The paper starts from a brief description of the most popular approaches of decision rules generating and the definition of oblique rule. Then, the *PCA* based decision rules induction is described. Afterwards, the results of experiments on the artificial and real data are described. The paper ends with some final conclusions and remarks.

2 Related Works

2.1 Decision Rules Generalisation

Decision rules are mathematical formulas given in the following form:

$$\text{IF } cond_1 \wedge cond_2 \wedge \ldots \wedge cond_n \text{ THEN } class = c$$

where $cond_i$ denotes the logical condition based on equality or inequality between the attribute value and some number or interval, and c is the label from the finite set of class labels. The main advantage of building classifiers on the basis of the decision rules is full interpretability of the rules. Sometimes it occurs that the rule description of the decision class consists of many rules which recognise very small number of objects. That leads to the postprocessing step that generalises and simplifies rules [15]. One of the most popular method of rules generalisation is rules shortening consisting of removal of elementary conditions on the basis of exhaustive searching or some heuristic methods. In the *RSES* (*Rough Set Exploration System*) [1] the shortening of the rule is performed as long as the rule quality remains on the same level.

The other approach of rules generalisation consists in joining two or more rules that describe the same class. The iterative algorithm described in [14] merges the ranges of conditions in corresponding elementary conditions (the elementary condition has the form $cond \equiv v(o) \in [a; b]$, where v is one of variables, o is object, $v(o)$ is the value of the v for the object o and a and b are ends of single range). Similar approach is presented in [8] - instead of the iterative way, rules are grouped before the merging step. In the paper [11,16] the complex elementary conditions in rules premises are introduced, they are linear combinations of attributes from elementary conditions of original rules premises.

2.2 Oblique Rules Induction

One of the main limitation of decision rules induction that are hyper-rectangles is their inability of describing the oblique shapes. If the class contains objects that are bounded with some hyperplane it may occur that this class will be described with a big amount of small rules: rules that recognise and describe only a small number of objects, rules that cover only a small part of the feature space. The solution is to define conditions on the basis of some hyperplane equation. Considering k-dimensional hyperplane H given with the equation:

$$H_1 x_1 + H_2 x_2 + \ldots + H_k x_k + H_{k+1} = 0$$

and the k-dimensional object o lets define $H(O)$ as:

$$H(O) = \sum_{i=1}^{k} H_i o_i + H_{k+1}.$$

Then the single conditional descriptor may take one of the following values: $H(O) \leq 0$ or $H(O) \geq 0$.

In this case it is assumed that each nonzero H_i is equivalent to one elementary condition.

There are also methods to generate this kind of rules from the results of other algorithms like from the oblique decision trees [4,7,10], from linear SVM [2], using the constructive induction [3,18], or directly from the data [9,12].

3 *PCA* Oblique Decision Rules

Previous approach to oblique rules generating (the ORG algorithm [9]) was based on semi-exhausting search of optimal parameters of separating hyperplanes. After some data transformation it was possible to determine boundaries for every hyperplane parameters. Then each combination of parameters (from minimal to maximal value with the defined step) was taken into consideration and evaluated. This gave quite satisfactory results but had a very high complexity. The usage of the global evaluation function also caused that the ORG did not recognise the situation when the class was build from several disjoint subclasses. In this approach each decision rule is built around the located subclass.

3.1 *PCA* Background

Principal component analysis (PCA) [17] is a statistical tool that transforms the coordinate system of the data in such a manner that the greatest variance of the data lies along the first coordinate, called the first principal component, the second greatest variance lies on the second coordinate, and so on. This transformation is useful in pattern description and data dimension reduction. However, the presented application concentrates on the abilities to find the coordinate system transformation only.

Having a data set X, whose mean is zero, it is possible to find the transformation using singular value decomposition or the covariance method. In the covariance method the covariance matrix of the data should be found

$$C = XX^T. \tag{1}$$

Using the covariance matrix the eigenvectors E_{vec} and corresponding to them eigenvalues E_{val} are calculated following the equation:

$$E_{vec}^{-1} C E_{vec} = E_{val}. \tag{2}$$

Next, the columns of E_{vec} are sorted in order of descending eigenvalues E_{val}. After this rearrangement the first eigenvector denotes the direction with the greatest variability, whereas the last one with the smallest one.

3.2 Finding Subclasses

Approximating the data with a hyper-cuboid introduces a noticeable error, as the data does not fill the object tight. Therefore, in order to diminish the error, it is suggested to divide the data into several subclasses and find separate hyper-cuboids for each of them. This solution removes the unnecessary space from the region belonging to the class and allows precisely defining the data boundaries.

Amongst many clustering algorithms one of the simplest was chosen: k-means. It has only one parameter to set – the number of objects that are nearest to the currently classified object. This application of the algorithm uses the adaptive way of finding the optimal value of subclasses number c. Let us assume the interesting range for $c_{min} \leq c \leq c_{max}$. Then for every c the division to the specified number of clusters is performed. The error metric is defined as an average distance from each cluster data to its cluster mean. The optimal value is defined taking into account two criteria: the smallest possible error value as well as the smallest cluster number.

3.3 Oblique Rules Induction

Subclass Rule Induction. The previous part of the article describes the way of finding subclasses for every class in the data. Having the set of objects that belong to the one subclass, it is assumed that their obliqueness can be described on the basis of the Principal Component Analysis.

Considering the $k-$dimensional subclass, after the PCA we obtain new k dimensional coordinate system which coordinate lines are parallel to PCA eigenvectors and go through the arithmetical centre of the subclass. Each eigenvector E_{vec_i} also determines the hyperplane $H(E_{vec_i})$ that is parallel to one of the new coordinate line and perpendicular to all the remaining ones.

In the presented approach it is assumed that every subclass of the data can be described with the oblique hyper-cuboid. The construction of the hyper-cuboid should fulfil the following conditions:

- every parallel pair of its sides is also parallel to one of the hyperplanes obtained from the PCA,
- all subclass objects lie between every pair of cuboid parallel sides,
- the distance between two parallel sides is the minimal.

The pair of parallel sides can be easily calculated from eigenvector E_{vec_i}:

$$H_{low_i} = E_{vec_i} V_{sm} \quad H_{high_i} = E_{vec_i} V_{gr} \tag{3}$$

where V_{sm} and V_{gr} are the smallest and the greatest values found in the data subclass transformed in the chosen direction.

Having hyperplanes that bound the $k-$dimensional subclass of the class c data we can build the decision rule R that is the logical conjunction of the following form:

$$R : \bigwedge_{i=1}^{k} H_{low_i}(P) \geq 0 \wedge H_{high_i}(P) \geq 0 \rightarrow class \ = \ c.$$

The point P is the point corresponding to the currently classified object.

A simple illustration for the single oblique rule induction is on the Fig. 1. Black dots represent the oblique data. Two dashed lines are coordinate lines in the new space, determined by the *PCA*. Solid lines are hyperplanes that bound the subclass. We may see that every hyperplane has at least one data point that belongs to the hyperplane. This assures that two parallel hyperplanes lie as close to themselves as possible and no data lies outside the hypercuboid.

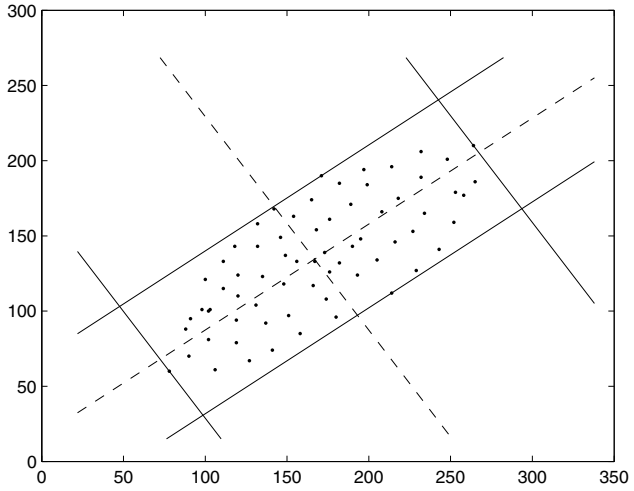


Fig. 1. Visualisation of the single *PCA* oblique rule induction

Orientation of the Hyperplane. Each hyperplane H divides the $k-$dimensional hyperspace $\mathbb{R}^k$ into two disjoint sets. The first of them contains points P that satisfy the condition $H(P) > 0$ and the second one contains points P that satisfy the following condition: $H(P) < 0$. The hyperplane H may be then interpreted as the boundary between these two subspaces. In presented approach it is assumed that the boundary is attached to the first set. Finally, it is claimed that each hyperspace H splits the hyperspace into two disjoint subsets:

$$H_+ = \{P \in \mathbb{R}^k : H(P) \geq 0\}$$

and

$$H_- = \{P \in \mathbb{R}^k : H(P) < 0\}$$

Now the notion of the hyperplane orientation is introduced. The point Q lies on the correct side of the hyperplane H iff $Q \in H_+$. In order to assure that all training data are on the correct side of the hyperplanes each subclass must check its hyperplanes orientation. If randomly selected point P lies on incorrect side of some hyperplanes then coefficients in their equations should be negated.

The Algorithm. Here the detailed description of *PCA* oblique decision rules induction is presented

1. Generate subclasses for each decision class in the adaptive way (assuming the minimal and the maximal number of subclasses)
2. For each subclass generate eigenvectors.
3. For each eigenvector generate two hyperplanes and check their orientation.
4. For each subclass build one decision rule from correctly oriented hyperplanes.

4 Experiments and Results

4.1 Experiments

Experiments were performed on the artificial and real data. Three sets of synthetic data are less known in the literature and comes from [16]. Each of them is the two-class problem with 1000 objects. Two sets are two-dimensional and classes are almost balanced (562:438 and 534:466). The third one is three-dimensional with unbalanced classes (835:165). Visualization of these data is shown on the Fig. 2.

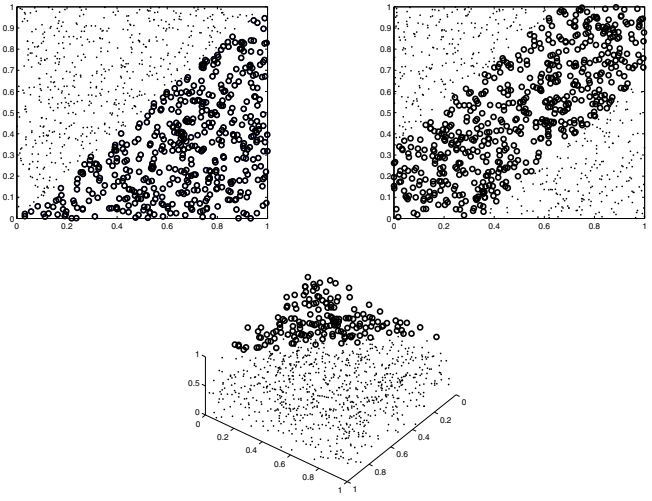


Fig. 2. Visualisation of the synthetic datasets: 2D (upper left); double2D (upper right); 3D (down)

Classes in the first set (2D) are divided by the diagonal of the square. In the second set (double2D) one class is placed in two opposite corners. The three-dimensional set (3D) with the unbalanced division contains the smaller class in the one of the corners of the cube. Results of experiments on these data are

shown in the Table 1. Other datasets (iris, balance, breast wisconsin) come from the *UCI* repository [5] and from [13] (Ripley). Results obtained on these sets are shown in the Table 2.

Experiments were performed with the 10-fold cross-validation. As the quality measure (in percents) the average of the rule precision and recall was used. The maximal number of subclasses per each decision class was set to 10.

The *PCA* oblique rules were compared with results of two other algorithms: *PART* and *ORG*. The Weka implementation of the *PART* was used [6]. Because Weka does not give information about the classification accuracy in each iteration of cross-fold validation, the information about its standard deviation cannot be presented. The *ORG* algorithm (*Oblique Rules Generator*) [9] searches for the best parameters of separating hyperplane in the grid of possible parameters values.

Table 1. Results on synthetic datasets

dataset	accuracy avg (std)			avg. rules number			avg. elem. cond. number		
	PART	*ORG*	*PCA ORG*	*PART*	*ORG*	*PCA ORG*	*PART*	*ORG*	*PCA ORG*
2D	95.5(-)	96.0(1.5)	98.4(1.0)	10	2	2	18	3	8
double 2D	93.8(-)	84.3(3.1)	99.0(1.1)	14	3	2	23	6	8
3D	94.8(-)	98.2(1.2)	98.3(1.6)	13	2	2	22	2	18

Table 2. Results on real datasets

dataset	accuracy avg (std)			avg. rules number			avg. elem. cond. number		
	PART	*ORG*	*PCA ORG*	*PART*	*ORG*	*PCA ORG*	*PART*	*ORG*	*PCA ORG*
iris	94(-)	94(4.6)	84(8.4)	2	3.1	3	3	5.2	32
balance	84(-)	92(2.4)	80(4.2)	46	6	3	126	12	32
Ripley	85(-)	81(8.4)	65(11.6)	4	2	6.6	6	4	8
breast w.	94(-)	97(1.7)	92(3.0)	10	3	2	33	19	162

4.2 Discussion

As it may be seen in the Table 1, that shows results of rules induction for some strongly oblique data, the *PCA* oblique rules induction algorithm gives the best (2D and double2D sets) or comparable (3D set) accuracy of classification. Even the number of rules is as small as possible: equal to the number of decision classes. The number of elementary condition may seem too big, but one should have that in mind that in the case of the *PCA* oblique rules this number is strongly dependent on the dimensionality of the input data. It is proportional to the squared number of dimensions: for each of d dimension, 2 hyperplanes with $d + 1$ parameters are generated. This is the cost of keeping rules interpretable because every single condition is just the linear combination of data attributes.

The Table 2, that shows results for real and common known data, points some aspects of the data, for which *PCA* oblique rules do not give the satisfactory results. One of the most important disadvantages of the presented method is its non-resistance for quite overlapping classes. It might be observed in the case of the iris data and in the case of Ripley's data, where the data characteristics has significant impact of the rules generation. For classes which distribution characteristic is similar (in the sense of *PCA*) rules inducted with the described method may become not satisfactory and valuable from the classification point of view. The visualisation of the two-class and two-dimensional Ripley's data is shown on the Fig. 3.

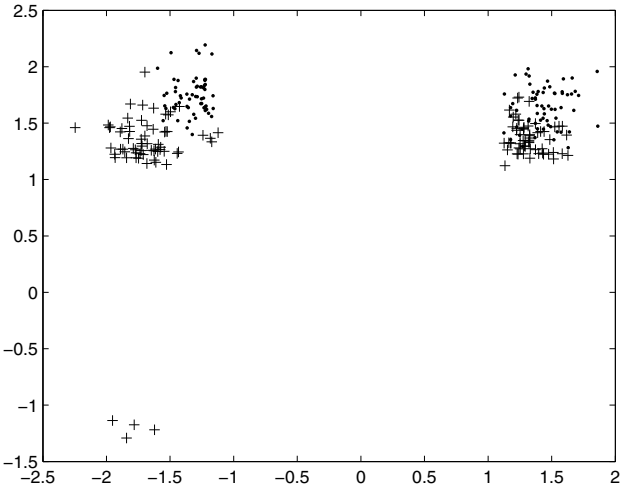


Fig. 3. Visualisation of the Ripley's data – problem with strong overlapping classes

What is worth to be noticed, the comparable accuracy of classification of balance and breast sets is achieved with smaller number of rules. It points that the presented algorithm can really generalise the class description in the oblique way without loosing its accuracy.

5 Conclusions and Further Works

In the paper the new algorithm of oblique decision rules induction is presented. The additional step aims to divide the data into subclasses, which allow to define hypercuboids defining class borders as tight to data as possible. In order to decide the hypercuboids sides direction the *PCA* analysis is applied. This division aims to enable generation of very precise oblique decision rules in second step.

The proposed algorithm was tested on artificial and real data set. The results proved that in most cases the number of generated rules is smaller in comparison to the *ORG* and *PART* algorithm. In case of the artificial data the results achieved with this algorithm proved better that the other approaches. However, when applying this algorithm to the real data, some drawbacks of this method occur, as it does not manage well with separate and overlapped classes.

Our further works will focus on decreasing the number of elementary conditions (which means that hyperplanes will be replaced with some hyperspaces) and also on limitation of number of hyperplanes (hyperspaces) in the single rule.

Acknowledgements. This work was supported by the European Union from the European Social Fund (grant agreement number: UDA-POKL.04.01.01-106/09).

References

1. Bazan, J., Szczuka, M.S., Wróblewski, J.: A New Version of Rough Set Exploration System. In: Alpigini, J.J., Peters, J.F., Skowron, A., Zhong, N. (eds.) RSCTC 2002. LNCS (LNAI), vol. 2475, pp. 397–404. Springer, Heidelberg (2002)
2. Bennett, K.P., Blue, J.A.: A support vector machine approach to decision trees. In: Proceedings of the IJCNN 1998, pp. 2396–2401 (1997)
3. Bloedorn, E., Michalski, R.S.: Data-Driven Constructive Induction. IEEE Intelligent Systems and Their Application 13(2), 30–37 (1998)
4. Cantu-Paz, E., Kamath, C.: Using evolutionary algorithms to induce oblique decision trees. In: Proc. of Genetic and Evolutionary Computation Conference, pp. 1053–1060 (2000)
5. Frank, A., Asuncion, A.: UCI Machine Learning Repository (2010), `http://archive.ics.uci.edu/ml`
6. Frank, E., Witten, I.H.: Generating Accurate Rule Sets Without Global Optimization. In: Proc. of the 15th International Conference on Machine Learning, pp. 144–151 (1998)
7. Kim, H., Loh, W.-Y.: Classification trees with bivariate linear discriminant node models. Journal of Computational and Graphical Statistics 12, 512–530 (2003)
8. Latkowski, R., Mikołajczyk, M.: Data Decomposition and Decision Rule Joining for Classification of Data with Missing Values. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Swiniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 299–320. Springer, Heidelberg (2004)
9. Michalak, M., Sikora, M., Ziarnik, P.: ORG - Oblique Rules Generator. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part II. LNCS, vol. 7268, pp. 152–159. Springer, Heidelberg (2012)
10. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research 2, 1–32 (1994)
11. Pindur, R., Sasmuga, R., Stefanowski, J.: Hyperplane Aggregation of Dominance Decision Rules. Fundamenta Informaticae 61(2), 117–137 (2004)
12. Raś, Z.W., Daradzińska, A., Liu, X.: System ADReD for discovering rules based on hyperplanes. Engineering Applications of Artificial Intelligence 17(4), 401–406 (2004)

13. Ripley, B.D.: Pattern Recognition and Neural Networks. Cambridge University Press (1996)
14. Sikora, M.: An algorithm for generalization of decision rules by joining. Foundation on Computing and Decision Sciences 30(3), 227–239 (2005)
15. Sikora, M.: Induction and pruning of classification rules for prediction of microseismic hazards in coal mines. Expert Systems with Applications 38(6), 6748–6758 (2011)
16. Sikora, M., Gudyś, A.: CHIRA – Convex Hull Based Iterative Algorithm of Rules Aggregation. Fundamenta Informaticae 123(2), 143–170 (2013)
17. Smith, L.I.: A tutorial on Principal Components Analysis (2002), http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf
18. Ślęzak, D., Wróblewski, J.: Classification Algorithms Based on Linear Combinations of Features. In: Żytkow, J.M., Rauch, J. (eds.) PKDD 1999. LNCS (LNAI), vol. 1704, pp. 548–553. Springer, Heidelberg (1999)

Cardinality Problem in Portfolio Selection

Penka Georgieva[1] and Ivan Popchev[2]

[1] Burgas Free University, Faculty of Engineering and Computer Science, Burgas, Bulgaria
penka.v.georgieva@hotmail.com
[2] Bulgarian Academy of Science, Sofia, Bulgaria
ipopchev@iit.bas.bg

Abstract. There is a variety of models for portfolio selection. However, in portfolio theory applications little or no attention is paid to the cardinality problem. In this paper, an algorithm for dealing with this problem is presented. The proposed allocation algorithm is implemented in a software system, which is based on the Fuzzy Logic Q-measure Model and manages financial investments in real time. Tests on real data from Bulgarian Stock Exchange are presented as illustration to the solution.

Keywords: portfolio selection, cardinality problem, fuzzy system, Q-measure.

1 Introduction to Cardinality Problem in Portfolio Theory

Let S be the capital to be invested in a financial portfolio and let $\{A_1, A_2, \ldots, A_j, \ldots, A_N\}$ be the set of available assets. If x_j is the quota of asset A_j in the portfolios, then the constraint for the sum of these quotas in optimization algorithms [2], [3], [4], [5], [8], [10], [11], [12], [13], [16], [17], [18] is:

$$\sum_{j=1}^{N} x_j = 1 \tag{1}$$

Apparently, this constraint cannot be precisely satisfied in the real world. Indeed, if P_j is the price of asset A_j and x_j are the quotas obtained after applying any algorithm for portfolio selection, then the number n_j of shares from asset A_j, included in the portfolio, is calculated as:

$$n_j = \left\lfloor \frac{x_j . S}{P_j} \right\rfloor \tag{2}$$

and thus the capital S_u used for constructing the portfolio is obtained as:

$$S_u = \sum_{j=1}^{N} n_j . P_j = S - \sum_{j=1}^{N} o_j \ , \tag{3}$$

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 208–217, 2013.

where $o_j = x_j.S - n_j.P_j$ are remainders due to (2). Then, the actual quota of each asset is $x_j^* = \frac{n_j.P_j}{S}$, and their sum is calculated as follows:

$$\sum_{j=1}^{N} x_j^* = \sum_{j=1}^{N} \frac{n_j.P_j}{S} = \frac{S - \sum_{j=1}^{N} o_j}{S} = 1 - \frac{\sum_{j=1}^{N} o_j}{S}. \quad (4)$$

Hence the component $\frac{\sum_{j=1}^{N} o_j}{S}$ in (3) changes the constraint (1) to the inequality:

$$\sum_{j=1}^{N} x_j \leq 1 \quad (5)$$

and so it modifies substantially any of the optimization algorithms. It is important to note that empirical tests show that the rate of $\sum_{j=1}^{N} o_j$ can reach up to 25% of *S*.

There are two main problems that arise at this point. Firstly, if the investor wants to use possible maximum of the capital *S*, then an assessment of $\sum_{j=1}^{N} o_j$ is needed and if the latter sum exceeds any of the prices P_j then the number n_j should be changed. Secondly, if short sales are excluded (as is the case on BSE) then (5) changes to:

$$\sum_{j=1}^{N} x_j \leq 1, \quad 0 \leq x_j \leq 1. \quad (6)$$

In this paper, one solution that satisfies (6) is proposed.

2 Description of the Software System

The software system is designed to manage financial data and investments (both individual and portfolio) in real time and is based on FLQM Model. The system consists of three modules (Fig.1):

- Module 1 – Data Managing Module (DMM);
- Module 2 – Q-Measure Fuzzy Logic Module (QFLM);
- Module 3 – Portfolio Construction Module (PCM).

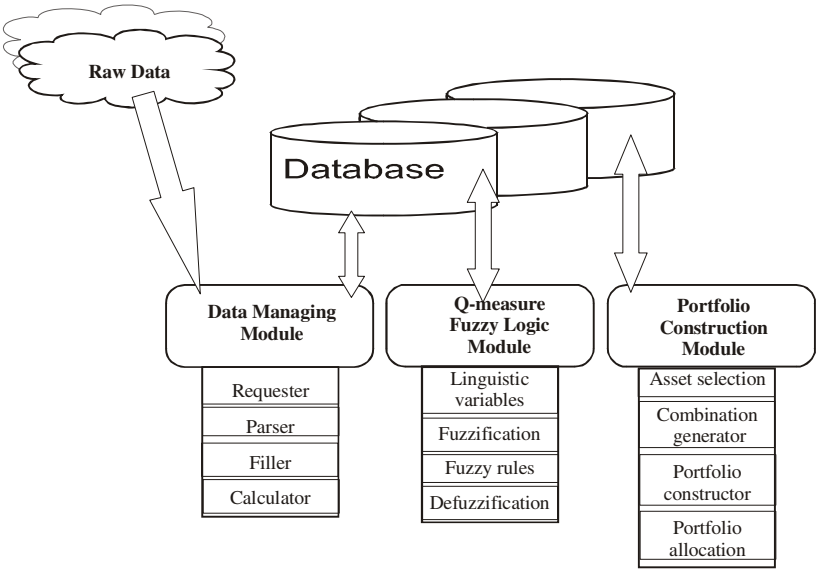


Fig. 1. A real time software system for financial portfolios selection

2.1 Data Managing Module

DMM is an application for collecting, storing and managing financial data in real time. In addition, in this module calculations of precise measurements of important asset characteristics (return, risk and *q*-ratio) are implemented.

Since the tests of the system are conducted over data from Bulgarian Stock Exchange, the information of interest (date, BSE code, open, close, high and low prices) is in the HTML code of www.bse-sofia.bg. The application is started automatically by Windows Task Scheduler. The **Requester** realizes the request to the web page of BSE (www.bse-sofia.bg). The **Parser** selects the needed data in a suitable form with respect to the next steps and therefore the HTML code of the page, downloaded by the **Requester**, is parsed with regular expressions. The **Filler** is the part of the application that deals with the missing data. The **Calculator** uses the information obtained by **Filler** and the following mathematical formulae to calculate the return, risk and *q*-ratio for each asset.

Let $P_1, P_2, ..., P_T$ be the sequence of daily prices. Then, asset return for day t is

$$r_t = \frac{P_t}{P_{t-1}}.$$

In time series of daily asset prices, there are days with no trading activity, i.e. there are missing data. One way to deal with this problem is to copy the last price a corresponding number of times, i.e.

$$\underbrace{P_{t-1}, P_{t-1}, \ldots P_{t-1}}_{k_t}, P_t .$$

Then geometric mean of returns is calculated as $r_g = \sqrt[k]{\prod_{t=2}^{T} r_t} = \sqrt[k]{\frac{P_T}{P_1}}$, where $k = \sum_{i=2}^{T} \Delta_t$ and Δ_t is the number of days between the non-missing observations at days $t-1$ and t, as decreased by 1. If log returns are used, the above considerations should be made very carefully because of the different number of days between the observations. Thus, if the return for the period Δ_t is $r_t^{\bullet} = \frac{r_t - 1}{\Delta_t} + 1$, then the log return at the moment t is $\ln\left(r_t^{\bullet}\right)$ and so the arithmetic mean of log returns is calculated:

$$\bar{r}^{\bullet} = \ln\left(\sqrt[T-1]{\prod_{t=2}^{T} r_t^{\bullet}}\right) = \frac{1}{T-1}\sum_{t=2}^{T} \ln r_t^{\bullet} .$$

Then, the annual norm of return *ANR* is

$$ANR = \left(e^{\bar{r}^{\bullet}} - 1\right).\frac{D}{k+1},$$

where *D* is the number of days in the financial year.

Finally, the annual return is

$$AR = ANR - 1. \tag{7}$$

In the proposed system, annual return is used as a measure of asset return as it is an adequate estimator for the change of the investment.

The commonly used idea for risk in investment theory is the variability of returns. The variability is calculated with different statistical tools, based on probability distributions but most often by the variance of the returns. [1], [9]

If log returns are used then the estimator of variance as arithmetic mean of log returns is calculated as follows:

$$s^2 = \frac{1}{T-2}\sum_{t=2}^{T}\left(\ln\left(r_t\right) - \bar{r_g}\right),, \tag{8}$$

and the *q*-ratio [15] equals the quotient of return and risk:

$$q = \frac{AR}{s}. \tag{9}$$

Since some investors have specific expectations about both the return and risk of their investments, there are two additional inputs in the system: a lower threshold of return r_0 and risk σ_0.

2.2 Q-measure Fuzzy Logic Module

QFLM is an application of Fuzzy Logic Q-measure Model [7]. Input data for this module are the crisp numerical values of asset characteristics, obtained by (7), (8) and (9) in DMM. These crisp values are fuzzified and after applying the aggregation rules a fuzzy variable Q-measure for each of the assets is derived. The output is a defuzzified crisp value of Q-measure. [14]

The linguistic variables are four: three input variables and one output variable. Input variables describe the characteristics of an asset: K_1 = {*return*}, K_2 = {*risk*} and K_3 = {*q-ratio*}. The output variable is Q = {*Q-measure*}.

The input variables K_1 = {*return*} and K_2 = {*risk*} consist of five terms each with corresponding parameters: *Very low* with Sigmoid membership function; *Low* with Gaussian membership function; *Neutral* with Gaussian membership function; *High* with Gaussian membership function, and *Very high* with Sigmoid membership function (Fig. 2).

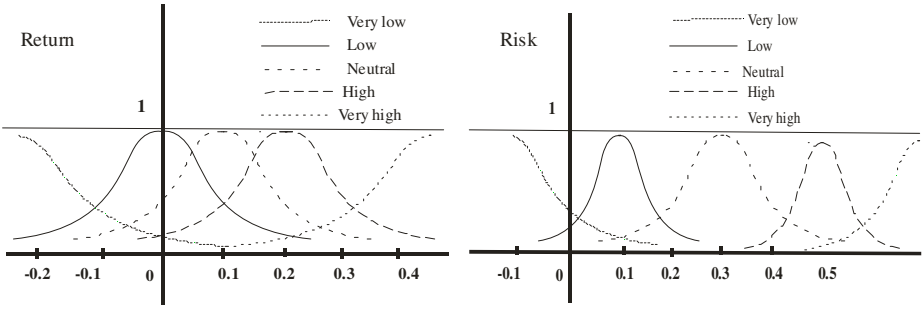


Fig. 2. Input linguistic variables K_1 = {*return*} and K_2 = {*risk*}

K_3 (Fig. 3) consist of three terms: *Small* with Sigmoid membership function; *Neutral* with Bell membership function, and *Big* with Gaussian membership function.

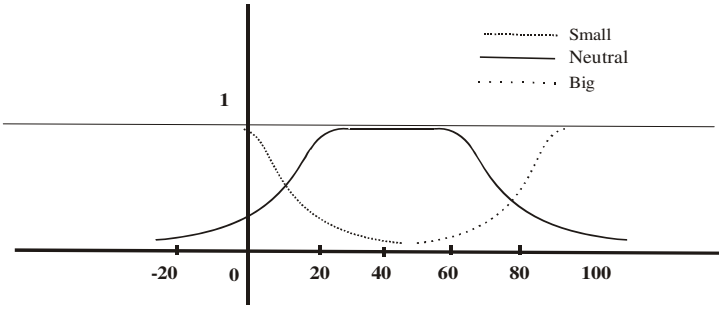


Fig. 3. Input linguistic variable $K_3 = q\text{-}ratio\}$

The output variable Q (Fig. 4) consists of five terms: *Bad*, *Not good*, *Neutral*, *Good* and *Very Good*, all with Gaussian membership functions.

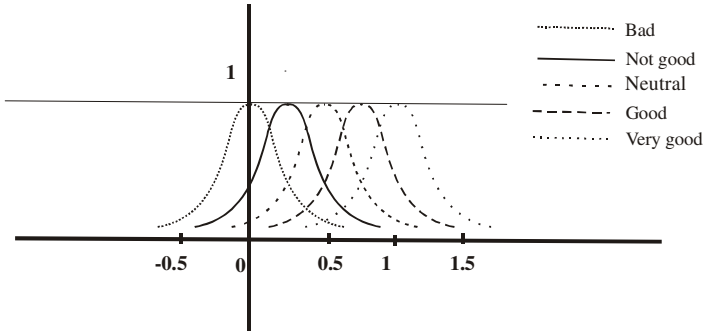


Fig. 4. Output linguistic variable *Q-measure*

All fuzzy rules in this module have the form:

IF $\{K_1$ is high$\}$ AND $\{K_2$ is low$\}$ $\{K_3$ is big$\}$ THEN (Q is good)

There are 24 fuzzy rules implemented in the system [6]. Although these rules adequately describe the most important possible situations that might arise in the process of investment decision-making, the list of fuzzy rules can be extended without changing the system's architecture.

As a defuzzification method, the method of centre of gravity has been chosen:

$$d_{CoG}(D) = \frac{\int_{-\infty}^{+\infty} x.D(x)dx}{\int_{-\infty}^{+\infty} D(x)dx},$$

and thus a crisp value for the asset quality is obtained as an output of QFLM.

The estimated value for each asset is then recorded in the database and used for managing individual assets or for constructing investment portfolios.

2.3 Portfolio Construction Module with Allocation Algorithm

In PCM, several portfolios are constructed. The investors' utility preferences are the key factor for choosing the optimal portfolio.

First, in Asset Selection all the assets are sorted in descending order by their Q-measure. Then several assets are selected with respect to a criterion set by the investor, e.g. maximum number of assets, or another. Here the case of max number of assets will be described. In the next step all possible combinations of these assets are generated and they form corresponding portfolios. Finally, if a substantial amount of capital is unused then a procedure for allocation takes place. This procedure modifies the portfolio to the optimal reduction of unused capital. Constructed portfolios are stored in a database and the investor can make a decision.

Now, if the investor wants no more than m assets in his portfolio, then the first m assets from the sort list are extracted. Then, these assets are combined in all possible ways to sets with 1, 2, ..., m elements. The number of these sets is $2^m - 1$, because the empty set is not taken into consideration. Now, for each of these sets a portfolio is constructed.

The share of asset A_j is denoted by x_j and equals

$$x_j = \frac{Q_j}{\sum_{j=1}^{n} Q_j},$$

where n is the number of financial assets in the portfolio, $n = 1, 2, \ldots, m$, and Q_j is the Q-measure of A_j.

Allocation Algorithm

The number of shares n_j and the amount of used capital are calculated according to (2) and (3). In case $S - S_u$ exceeds a given percentage of the initial capital an allocation procedure takes place. In the first step of this procedure, the remaining capital is compared to the price of the asset with the highest Q-measure. If $S - S_u$ is bigger than that price, then additional number of shares of this asset is added to the portfolio. Otherwise, the comparison is repeated for the next item in the sort list. This process repeats until the remaining capital is small enough and no more shares can be bought. At each step, the number of additional shares n_{j_a} of an asset is

$$n_{j_a} = \left\lfloor \frac{x_j.(S - S_u)}{P_j} \right\rfloor.$$

Finally, portfolio return R_p, risk σ_p and q-ratio q_p are calculated for all $2^m - 1$ portfolios:

$$R_p = \sum_{j=1}^{N} x_j . r_j \quad , \qquad \sigma_p = \sum_{j=1}^{N} x_j . s_j \quad , \qquad q_p = \frac{R_p}{\sigma_p} .$$

Now, each portfolio is put through QFLM in order to obtain its Q-measure. The portfolios with their characteristics are stored in the data base.

3 Experiments and Results

To study the relationship between the size of the investment capital and the need to implement additional allocation procedure, various tests have been conducted. In this paper, portfolios of seven assets and initial capital of S = BGN 1 000 have been constructed with assets listed on BSE on 20.06.2012 to illustrate the above considerations. It is important to note that the additional allocation procedure does not apply if the unused capital is less than 0.5% of the initial capital. This percentage is set up to avoid transaction costs, whenever unused capital is relatively small.

The initial portfolio has return R_p = 1.78318324, risk σ_p = 0.01519687, Q-measure = 0.81230537 and in this case $\sum_{j=1}^{N} o_j$ = BGN 66.58 (Table 1).

Originally, the portfolio is constructed with a high percentage of unused capital (6.658%, which is well over 0.5%). To reduce this rate, the allocation procedure is applied.

Table 1. Initial portfolio with 7 assets, S_u = BGN 66.58

BSE code	P_j	n_j	x_j	Risk	Annual norm of return	Q-measure
3JU	45	3	0.14538565	0.0241193	1.99408284	0.82919283
5ORG	90	1	0.14245006	0.00247399	1	0.81245
6A6	1.74	81	0.14244872	0.01237877	1.85524372	0.81244236
BLKC	0.471	302	0.14244326	0.02221776	2.52884615	0.81241119
4EC	1.91	74	0.14243653	0.02754392	2.86830926	0.81237285
SO5	1.846	77	0.14243505	0.00874433	1.18275862	0.81236439
5BD	0.754	188	0.14240073	0.00871467	1.04851752	0.81216861

The portfolio after the first allocation procedure has return R_p = 1.79868443, risk σ_p = 0.01556071, Q-measure = 0.81228124 and $\sum_{j=1}^{N} o_j$ = BGN 15.97 (Table 2). It can be noted that the risk of this portfolio is less than the risk of the initial portfolio, the return is higher and the Q-measure is slightly smaller. The unused capital is reduced to BGN 15.97, but an additional allocation procedure is still possible.

Table 2. Portfolio with 7 assets after first and second allocation

	After first allocation			After second allocation		
BSE code	P_j	n_j	x_j	P_j	n_j	x_j
3JU	45	3	0.135	45	3	0.135
5ORG	90	1	0.09	90	1	0.09
6A6	1.74	87	0.15138	1.74	88	0.15312
BLKC	0.471	325	0.153075	0.471	331	0.155901
4EC	1.91	79	0.15089	1.91	80	0.1528
SO5	1.846	82	0.151372	1.846	83	0.153218
5BD	0.754	202	0.152308	0.754	206	0.155324

The portfolio after the second allocation has $R_p = 1.81988325$, risk $\sigma_p = 0.01574007$, Q-measure = 0.81228817 and $\sum_{j=1}^{N} o_j$ = BGN 4.64. (Table 2)

In this example, the unused capital is reduced from 6.658% to 0.464% of the investment capital as a result of applying the allocation procedure twice. The last portfolio has a very good Q-measure, which is an indicator for stable behaviour in future.

Clearly, the allocation procedure can be repeated as many times as needed.

4 Conclusions

In this paper, a solution to the cardinality problem in portfolio theory is presented. This solution is obtained after applying an allocation procedure in a real time software system for managing individual assets and portfolio investments. This system is based on the Q-measure of an asset. The Q-measure incorporates return, risk and their ratio, and being modeled with fuzzy logic tools it intuitively reflects the process of investment decisions in economic environment with enormous amount of data, which is often incomplete and imprecise. The system provides a procedure for portfolio allocation that aims at maximal possible use of investment capital. Although it is not based on an optimization algorithm it solves the cardinality problem in portfolio management to a degree that suits investors.

Acknowledgement. The paper was partially financed by Burgas Free University under project Modeling Laboratory'13.

References

1. Campbell, J., Lo, A., MacKinlay, C.: The Econometrics of Financial Markets. Princeton University Press, Princeton (1996)
2. Carlsson, C., Fuller, R., Giove, S.: Optimization under Fuzzy Rule Constraints. In: Proceedings of Eurofuse-SIC 1999, Budapest (1999)
3. Craig, W., French: The Treynor Capital Asset Pricing Model. Journal of Investment Management 1(2), 60–72 (2003)
4. Elton, E.J., Gruber, M.J., Padberg, M.W.: Simple Criteria For Optimal Portfolio Selection. Journal of Finance 31(5), 1341–1357 (1976)
5. Fang, Y., et al.: Fuzzy Portfolio Optimization. Springer (2008)
6. Georgieva, P., Popchev, I.: Application of Q-measure in Real Time Fuzzy System for Managing Financial Assets. IJSC 3(4), 21–38 (2012)
7. Georgieva, P., Popchev, I.: Fuzzy Logic Q-measure Model for Managing Financial Investments. Compus Rendus Acad. Bulg. Sci. (2013)
8. Haugen, R.A.: Modern Investment Theory. Prentice-Hall, NJ (2000)
9. Judge, G., Hill, R.C., Griffiths, W.E., et al.: Introduction to the Theory and Practice of Econometrics. Wiley, New York (1988)
10. Konno, H., Suzuki, K.: A meanvariance-skewness portfolio optimization model. Journal of the OperationsResearch Society of Japan 38(2), 173–187 (1995)
11. Mansini, R., Speranza, M.G.: Heuristic algorithms for the portfolio selection problem with minimum transaction lots. European Journal of Operational Research 114, 219–233 (1999)
12. Markowitz, H.: Portfolio Selection. The Journal of Finance 7(1), 77–91 (1952)
13. Nawrocki, D.: Portfolio Analysis with a Large Universe of Assets. Applied Economics 28 (1996)
14. Peneva, V., Popchev, I.: Fuzzy multi-criteria decision making algorithms. Compus Rendus Acad. Bulg. Sci. 63(7) (2010)
15. Popchev, I., Georgieva, P.: Fuzzy Approach for Solving Multicriteria Investment Problems. Innovative Techniques in Instruction Technology, ITITEEE, 427–431 (2008)
16. Schaerf, A.: Local Search Techniques for Constrained Portfolio Selection Problems. Computational Economics 20, 177–190 (2002)
17. Sharp, W.: Capital asset prices: A theory of market equilibrium under conditions of risk. Journal of Finance 19(3), 425–442 (1964)
18. Ivanova, Z., Stoilova, K., Stoilov, T.: Portfolio Optimization-Information Service in Internet. Academic Publ. House Marin Drinov (2005) (in Bulgarian)

Full and Semi-supervised k-Means Clustering Optimised by Class Membership Hesitation

Piotr Płoński and Krzysztof Zaremba

Institute of Radioelectronics, Warsaw University of Technology,
Nowowiejska 15/19,00-665 Warsaw, Poland
{pplonski,zaremba}@ire.pw.edu.pl

Abstract. K-Means algorithm is one of the most popular methods for cluster analysis. K-Means, as the majority of clustering methods optimise clusters in an unsupervised way. In this paper we present a method of cluster's class membership hesitation, which enables k-Means to learn with fully and partially labelled data. In the proposed method the hesitation of cluster during optimisation step is controlled by Metropolis-Hastings algorithm. The proposed method was compared with state-of-art methods for supervised and semi-supervised clustering on benchmark data sets. Obtained results yield the same or better classification accuracy on both types of supervision.

Keywords: k-Means, Semi-supervised clustering, Supervised clustering, Classification, Metropolis-Hastings algorithm.

1 Introduction

Cluster analysis is one of the most used data mining techniques [6]. Clustering assigns similar data points into the same cluster, whereas separate different data points by assigning them to different clusters. Clustering is NP-hard problem and among clustering algorithms the most popular is k-Means [14], [10]. It is a simple algorithm that requires tuning of the k number of clusters and selecting optimal distance metric. There have been proposed many heuristic improvements to find optimal k [7], [13] and to designate distance metric [20], [5], [16]. The drawback of k-Means algorithm is its sensitivity to initial cluster centre values. Therefore, a careful seeding of k-Means is needed [3].

K-Means is originally an unsupervised learning algorithm. However, there were proposed several techniques to use it in a supervised and semi-supervised manner. They can be divided into three groups. The first group of methods upon labelled data is optimising the distance metric, which gives small distance for samples from the same class and separates by a large distance samples from different classes. After metric optimisation step, the k-Means in an unsupervised way is performed with learned metric [1], [20]. The other group of methods uses labelled data to compute initial centre values to k-Means algorithm [22], [4]. The last group of methods used labelled data to generate constrains. They control which points can be in the

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 218–225, 2013.

same cluster and which should be separated by assigning into different clusters. The constrains are used during k-Means learning [21], [5].

Herein, we present the method for controlling supervision process in k-Means. It is based on Metropolis-Hastings (MH) algorithm [15], [8] well known from Simulated Annealing (SA) method [11]. MH algorithm is used to simulate a hesitation of cluster's class membership during k-Means learning. In the supervision process both fully and partially labelled data can be used. SA was already used in k-Means algorithm focused on seed selection [17] or parameters tuning [23] rather than supervision control. Recently, we proposed a similar method for controlling learning of neurons in Self-Organising Maps [19].

2 Methods

Let's denote data set as $D = \{(\boldsymbol{x}_i, c_i)\}$, where $\boldsymbol{x}_i$ is an attribute vector, $\boldsymbol{x} \in \mathcal{R}^d$ and c_i is a discrete class number of i-th sample, $i = [1, 2, ..., N]$ and $c = [1, 2, ..., C]$.

2.1 K-Means Algorithm

K-Means is an unsupervised learning algorithm. However, it can be used for classification. The two methods that enable handling class labels in k-Means are described below. The original K-Means algorithm can be described in four steps:

1. Initialize k cluster's centre $\{\mu_1, \mu_2, ..., \mu_k\}$ with randomly selected values $\boldsymbol{x}_i \in D$.
2. Assign each data point $\boldsymbol{x}_i$ to the closest cluster ζ_h, $h = \underset{h}{\operatorname{argmin}} ||\boldsymbol{x}_i - \mu_h||^2$.
3. Update each cluster centre μ_i as mean of the assigned points, $\mu_i = 1/|\zeta_i| \sum_{\boldsymbol{x}_i \in \zeta_i} \boldsymbol{x}_i$.
4. Repeat steps 2 and 3 until convergence.

In k-Means algorithm information about sample's class label is not used. However, after unsupervised learning for each cluster can be assigned class label based on major vote of sample's class, which belongs to the cluster. In testing phase, for input sample is designated a class label from the closest cluster. We will call this method vote-k-Means.

Another method for using unsupervised k-Means for classification is to assign clusters to the classes arbitrarily - usually the same number of clusters for each class. During learning for each sample only clusters with the same class as the sample's class are considered [9]. Therefore, clusters are updated only with samples from the same class. For a testing sample a label the same as class of the closest cluster is given. We will call this method a class-k-Means.

2.2 Seeded k-Means

The next method that uses labels is seeded-k-Means [4]. It uses labelled data to compute better initial values of cluster centres. In seeded-k-Means we assume

that as algorithm input we have k disjoint sets $S = \{S_1, S_2, ..., S_k\}$, $S \subseteq D$, on which supervision is provided - for each $\boldsymbol{x}_i \in S$ is known a cluster ζ_l to which it belongs. The cluster centres are initialized as follows:

$$\mu_i = \frac{1}{|S_i|} \sum_{\boldsymbol{x}_j \in S_i} \boldsymbol{x}_j. \tag{1}$$

In [4] there is also an assumption that each cluster has at least one seed point. However, this condition is hard to be satisfied, and for clusters without any seed point a randomly selected value $\boldsymbol{x}_i$ can be used [22].

2.3 Proposed Method (MH-k-Means)

In the proposed method, for each cluster we compute a class membership in each iteration. Let's note it as $P_l(j)$, where l is cluster index and j is class number. For each sample a group of clusters is selected, which will take part in the closest cluster finding. Selection is described by a matrix T, where $T_l^i = 1$ means that cluster ζ_l will participate in learning, using i-th sample, $T_l^i = 0$ otherwise. Clusters for training are selected in two steps. At first we choose clusters having maximum probability for the class matching the class c_i of the input sample:

$$T_l^{i(1)} = \begin{cases} 1 & \text{if } \arg\max_j(P_l(j)) = c_i; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In the second step, remaining clusters with $T_l^{i(1)} = 0$ are considered. The decision on joining into the training with i-th sample is taken upon MH algorithm. The probability of joining is computed upon the following equation:

$$J_l^i = 1 - exp(-\rho P_l(c_i) t_{stop}/t), \tag{3}$$

where ρ is the parameter that controls the intensity of hesitation, $\rho \in [0, 1]$. The greater ρ, the more clusters are selected additionally to learning in the MH step. In eq.(3) the parameter t is a number of current algorithm iteration and t_{stop} is an overall number of algorithm iterations. The fact that t is presented in (3) ensures that clusters added during MH step will be selected less frequently at the end of learning process than at its beginning. This can be interpreted as a hesitation of the cluster, which decreases during the learning. To decide whether the MH decision is positive, we draw a random number a from a uniform distribution, $a \in [0, 1]$. The cluster will be added to the training group if a is smaller than J_l^i:

$$T_l^{i(2)} = \begin{cases} 1 & \text{if } a < J_l^i; \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

This procedure is repeated for each sample. We called $T_l^{i(2)} = 1$ as a positive MH decision. The final decision on cluster selection is a logical 'or' of the decisions $T_l^i = T_l^{i(1)} \vee T_l^{i(2)}$. For i-th sample the closest cluster ζ_h is computed as follows:

$$h = \underset{h}{\operatorname{argmin}} ||\boldsymbol{x}_i - \mu_h||^2 \wedge T_h^i = 1. \tag{5}$$

After the presentation of all samples new class membership probabilities are computed for each cluster:

$$P_l(h) = \frac{|\boldsymbol{x}_i \in \zeta_l \wedge \boldsymbol{x}_i \in c_h|}{|\boldsymbol{x}_i \in \zeta_l)}. \tag{6}$$

The procedure described above is repeated till algorithm's convergence. We have called the proposed algorithm MH-k-Means. The MH-k-Means algorithm described above works on fully labelled data. In case of partially labelled data, for samples without class label we assume that $T_h^i = 1$ for all clusters. Thus, all clusters participate in training. For labelled samples the procedure described above is used.

3 Results

To test performance of MH-k-Means method on fully labelled data, we will compare it to the Learning Vector Quantization algorithm (LVQ) [12], vote-k-Means, class-k-Means and seeded-k-Means. On partially labelled data sets, we will compare MH-k-Means to seeded-k-Means and vote-k-Means[1]. The comparison is made on 6 real data sets. We used data sets 'Wine', 'Ionosphere', 'Iris', 'Sonar', 'Spam' from the 'UCI Machine Learing Repository' [2] [2], and set 'Faces' are from the 'The ORL Database of Faces'[3]. Data sets are described in Table 1.

Table 1. Description of data sets used to test performance, number of clusters used to each data set and optimal ρ in MH-k-Means. (*In 'Faces' data set, the number of attributes was reduced with PCA.)

	Train examples	Test examples	Attributes	Classes	# clusters	MH ρ
Faces	320	80	50*	40	80	1
Sonar	166	42	60	2	36	0.5
Spam	3680	921	57	2	72	0.25
Iris	120	30	4	3	12	1
Ionosphere	280	71	34	2	24	0.25
Wine	142	36	13	3	12	1

In all experiments we train algorithms with number of iterations $t_{stop} = 200$. For LVQ we use learning rate $\eta_1 = 0.1$, exponentially decreasing to $\eta_{200} = 0.001$. All algorithms, except seeded-k-Means, were initialized with random samples. For seeded-k-Means we assigned available labelled samples to appropriate clusters reusing clustering from class-k-Means, in case of incomplete seeding of the cluster we used initialization with random sample [22]. For each data set, we arbitrarily

[1] We used only labelled data for cluster's class voting.

[2] `http://archive.ics.uci.edu/ml/`

[3] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

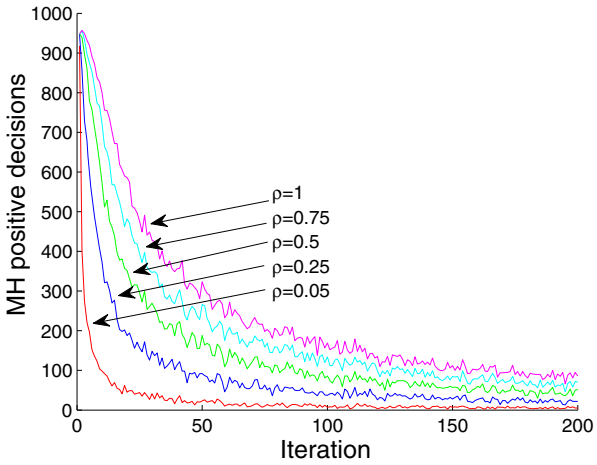


Fig. 1. Number of positive MH decisions in MH-k-Means algorithm taken in each training iteration for different ρ values on 'Iris' data set

Table 2. Percent of incorrect classification on fully labelled testing subsets. Results are mean and σ over 10 runs.

	LVQ	vote-k-Means	class-k-Means	seeded-k-Means	MH-k-Means
Faces	8.25±3.34	28.0±4.5	6.5±3.72	6.88±4.09	**4.0±2.34**
Sonar	**14.52±7.48**	26.9±8.1	15.48±4.92	17.38±5.73	**14.52±5.32**
Spam	**13.34±1.16**	18.43±1.25	14.18±1.77	17.13±2.19	14.47±1.3
Iris	4.0±2.11	3.33±3.51	4.33±2.74	4.67±3.58	**3.0±1.89**
Ionosphere	10.99±2.95	9.58±3.31	12.82±2.61	10.0±3.22	**8.73±2.38**
Wine	5.0±3.66	7.5±3.72	4.72±3.72	6.39±5.08	**4.44±2.68**

chose the cluster number (selecting optimal cluster size is not in the scope of this paper), selected values are presented in Table 1. The total cluster number for each algorithm type is equal. For MH-k-Means the parameter ρ must be tuned. We checked several values of ρ, $\rho = \{0.05, 0.25, 0.5, 0.75, 1\}$ and for each data set an optimal value was selected by cross-validation. Selected ρ values are presented in Table 1. The impact on number of positive MH decision, depending on ρ value is demonstrated on 'Iris' set in the Fig.1. The greater ρ value is, the more positive MH decisions are made and the more frequently cluster takes part in training with the sample, of which class is different than its major class. For each data set we made 10 repetitions to avoid effect of local minima, each time training and testing subsets were redrawn. As an accuracy measure of clustering, we take a percentage of incorrect classifications - for each testing sample we compute the closest cluster and compare the labels. The mean results on testing subsets for all the methods on fully labelled sets are presented in Table 2.

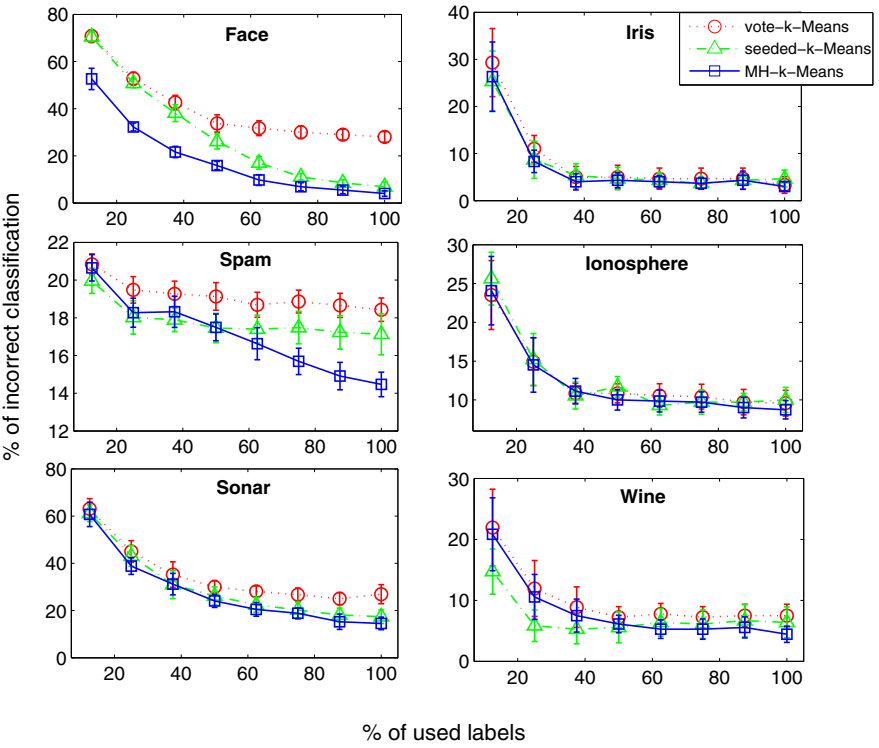


Fig. 2. Performance of vote-k-Means, seeded-k-Means and MH-k-Means on partially labelled data. Results are mean and σ over 10 runs.

With one exception the MH-k-Means achieves lower error rates than other methods. On 'Spam' data sets the best method was LVQ. The MH-k-Means obtained the greatest improvement over 'vote-k-Means' on 'Faces' data set. There are 40 classes (persons) in 'Faces' set, therefore it is difficult to compute good clusters without any information about class labels during learning. On this set, MH-k-Means uses the largest ρ, which means that cluster's class membership hesitates the most. On 'Sonar' data set, LVQ and MH-k-Means gain the same average of incorrect classification. However, MH-k-Means has lower standard deviation value. On 'Iris' set all methods give similar results, with slightly lower error of MH-k-Means. On this set and 'Ionosphere' the vote-k-Means accuracy is better than LVQ. They are simple sets, therefore without any supervision a good minima can be obtained. On 'Wine' set, class-k-Means is better than LVQ. The poorest accuracy on all data sets was obtained by vote-k-Means method. This was expected, as this method does not use the information about sample's class during the cluster's centre tuning. Seeded-k-Means have results slightly worse than class-k-Means, on all sets, except 'Ionosphere'. This is due to the fact that clusters obtained by class-k-Means were used in initialization of seeded-k-Means.

To test performance of the proposed MH-k-Means method on partially labelled data, we used only part of available labels in training subsets, in per cent $r = \{12.5, 25, 37.5, 50, 75, 87.5, 100\}$. The results of comparison are presented in Fig.2. The MH-k-Means is significantly better than other methods on 'Faces', 'Spam' and 'Sonar' data sets. On 'Iris', 'Ionosphere' and 'Wine' all methods seem to give similar results. The 'Faces', 'Spam' and 'Sonar' seem to be more complex in classification than other sets, therefore we observe that using information about class labels during learning gives better clustering. For simple data sets semi-supervised and unsupervised methods gain similar minima.

4 Conclusions

We present a novel method MH-k-Means for learning k-Means with fully and partially labelled data. In each iteration for every cluster a class membership is computed. Upon this, for each sample for the closest cluster finding a group of clusters with the same as sample's class is selected. What is more, the hesitation mechanism is introduced, which enables clusters with different class to take part in the closest cluster finding. The hesitation is based on Metropolis-Hastings algorithm, with hesitation intensity controlled by ρ parameter and current iteration number. The number of MH positive decisions decrease during learning, which can be interpreted as making clusters more confident. In case of partially labelled data, the clusters selection for learning is made only for labelled samples. For unlabelled samples all clusters participate in training. The proposed MH-k-Means method was compared to other state-of-art methods on classification tasks. The results confirm that proposed method obtains better or similar accuracy than other methods. Matlab implementation of the MH-k-Means algorithm is available at `http://home.elka.pw.edu.pl/~pplonski/mh_kmeans`. Future work will be focused on two aspects: testing what impact on MH-k-Means accuracy has the noise of class labels and the use of distance metric learning method for classification accuracy improvement[18].

References

1. Al-Harbi, S.H., Rayward-Smith, V.J.: Adapting k-means for supervised clustering. Applied Intelligence 24, 219–226 (2006)
2. Asuncion, A., Newman, D.J.: UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences (2007)
3. Arthur, D., Vassilvitskii, S.: K-means++: The Advantages of Careful Seeding. In: Symposium on Discrete Algorithms (2007)
4. Basu, S., Banerjee, A., Mooney, R.J.: Semi- supervised clustering by seeding. In: Proceedings of the 19th International Conference on Machine Learning, pp. 19–26 (2002)
5. Bilenko, M., Basu, S., Mooney, R.J.: Integrating constraints and metric learning in semi-supervised clustering. In: Proceedings of the 21th International Conference on Machine Learning, pp. 81–88 (2004)

6. Du, K.-L.: Clustering: A neural network approach. Neural Networks 23, 89–107 (2010)
7. Hamerly, G., Elkan, C.: Learning the k in k-means. In: Neural Information Processing Systems Conference (2003)
8. Hastings, W.K.: Monte Carlo Sampling Methods Using Markov Chains and Their Applications. Biometrika 57, 97–109 (1970)
9. Hinton, G.E., Dayan, P., Revow, M.: Modeling the manifolds of images of handwritten digits. IEEE Transactions on Neural Networks 8, 65–74 (1997)
10. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern Recognition Letters 31, 651–666 (2010)
11. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. Science 220, 671–680 (1983)
12. Kohonen, T.: The Self-Organizing Map. Proceedings of the IEEE 78, 1464–1480 (1990)
13. Likas, A., Nikos, V., Verbeek, J.J.: The global k-means clustering algorithm. Pattern Recognition 36, 451–461 (2003)
14. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceeding of the 5th Berkeley Symposium, pp. 281–297 (1967)
15. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of State Calculations by Fast Computing Machines. Journal of Chemical Physics 21, 1087–1092 (1953)
16. Olszewski, D.: Asymmetric *k*-Means Algorithm. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 1–10. Springer, Heidelberg (2011)
17. Perim, G.T., Wandekokem, E.D., Varejão, F.M.: K-Means Initialization Methods for Improving Clustering by Simulated Annealing. In: Geffner, H., Prada, R., Machado Alexandre, I., David, N. (eds.) IBERAMIA 2008. LNCS (LNAI), vol. 5290, pp. 133–142. Springer, Heidelberg (2008)
18. Płoński, P., Zaremba, K.: Improving Performance of Self-Organising Maps with Distance Metric Learning Method. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part I. LNCS, vol. 7267, pp. 169–177. Springer, Heidelberg (2012)
19. Płoński, P., Zaremba, K.: Self-Organising Maps for Classification with Metropolis-Hastings Algorithm for Supervision. In: Huang, T., Zeng, Z., Li, C., Leung, C.S. (eds.) ICONIP 2012, Part III. LNCS, vol. 7665, pp. 149–156. Springer, Heidelberg (2012)
20. Siriseriwan, W., Sinapiromsaran, K.: Attributes Scaling for K-means Algorithm Controlled by Misclassification of All Clusters. In: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, pp. 220–223 (2010)
21. Wagstaff, K., Cardie, C., Rogers, S.: Constrained k-means clustering with background knowledge. In: Proceedings of the 18th International Conference on Machine Learning, pp. 577–584 (2001)
22. Wang, X., Wang, C., Shen, J.: Semi–supervised K-Means Clustering by Optimizing Initial Cluster Centers. In: Gong, Z., Luo, X., Chen, J., Lei, J., Wang, F.L. (eds.) WISM 2011, Part II. LNCS, vol. 6988, pp. 178–187. Springer, Heidelberg (2011)
23. Yang, W., Rueda, L., Ngom, A.: A Simulated Annealing Approach to Find the Optimal Parameters for Fuzzy Clustering Microarray Data. In: Proceedings of the 25th International Conference of Chilean Computer Science Society, pp. 45–55 (2005)

Defining Semantic Meta-hashtags for Twitter Classification

Joana Costa[1,2], Catarina Silva[1,2], Mário Antunes[1,3], and Bernardete Ribeiro[2]

[1] School of Technology and Management,
Computer Science Communication and Research Centre
Polytechnic Institute of Leiria, Portugal
{joana.costa,catarina,mario.antunes}@ipleiria.pt
[2] Department of Informatics Engineering,
Center for Informatics and Systems of the University of Coimbra (CISUC), Portugal
{joanamc,catarina,bribeiro}@dei.uc.pt
[3] Center for Research in Advanced Computing Systems (CRACS), Portugal
mantunes@dcc.fc.up.pt

Abstract. Given the wide spread of social networks, research efforts to retrieve information using tagging from social networks communications have increased. In particular, in Twitter social network, *hashtags* are widely used to define a shared context for events or topics. While this is a common practice often the *hashtags* freely introduced by the user become easily biased. In this paper, we propose to deal with this bias defining semantic meta-hashtags by clustering similar messages to improve the classification. First, we use the user-defined *hashtags* as the Twitter message class labels. Then, we apply the meta-hashtag approach to boost the performance of the message classification.

The meta-hashtag approach is tested in a Twitter-based dataset constructed by requesting public *tweets* to the Twitter API. The experimental results yielded by comparing a baseline model based on user-defined *hashtags* with the clustered meta-hashtag approach show that the overall classification is improved. It is concluded that by incorporating semantics in the meta-hashtag model can have impact in different applications, e.g. recommendation systems, event detection or crowdsourcing.

Keywords: Meta-hashtags, Semantic, Text Classification, Twitter.

1 Introduction

Twitter is a social media platform that provides a microblogging service where users are able to post text-based messages of up to 140 characters, also known as *tweets*. It can also be considered an online social network, as users can link themselves by defining others to follow, and consequently have their own followers. The underlying concept of Twitter is to share the everyday activities with friends and family in a simple way. However, *tweets* may contain information of broad interest [1] and have a wide range of applications and uses, like event detection [2–5], academic tool [6–8], news media [2,9] or mining political opinion [10,11].

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 226–235, 2013.

Twitter provides the possibility of including an *hashtag*, a single word started with the symbol "#" , in order to classify the content of a message and improve search capabilities. This can be particularly important considering the amount of data produced in Twitter social network. Besides improving search capabilities, *hashtags* have been identified as having multiple and relevant potentialities, like promoting the phenomenon described in [12] as *micro-meme*, i.e. an idea, behavior or style that spreads from person to person within a culture [13]. By tagging a message with a trending topic hashtag, a user expands the audience of the message, compelling more users to express their feelings about the subject [14].

Considering the importance of the hashtag in Twitter, it is relevant to study the possibility of evaluating message contents in order to predict its hashtag. If we can classify a message based on a set of *hashtags*, we are able to suggest an hashtag for a given *tweet*, bringing a wider audience into discussion [15], spreading an idea [16], get affiliated with a community [17], or bringing together other Internet resources [18].

We propose an approach to deal with the bias resulting from the freely user-defined *hashtags*, by defining semantic meta-hashtags to identify clusters of similar messages, in order to improve their classification. First, we use the user-defined *hashtags* as the Twitter message class labels. Then, we define meta-hashtags by grouping the most used *hashtags* and their related *hashtags* into a meta-class and applied the meta-hashtag approach to boost the performance of the initial message classification. Both the initial model and the meta-hashtag model were tested with the initial user defined *hashtags* and the results are presented by comparing the classification performances obtained.

The rest of the paper is organized as follows. We start in Section 2 by describing the related work regarding social networks and meta-class approaches. We then proceed in Section 4 to explain the experimental setup, including the dataset description, the pre-processing methods, learning and evaluation approaches. In Section 5 we present and analyse the results obtained. Finally, in Section 6 we present the most relevant conclusions and delineate some directions for future work.

2 Related Work

Social networks have gained significant importance and are being widely studied in many fields in the last years. Modern challenges in social networks involve not only computer science matters but also social, political, business, and economical sciences. In computer science, and considering our focus on Twitter, recent works comprise event detection [3, 4], information spreading [19], community mining [20], crowdsourcing [21] and sentiment analysis [11].

Regarding Twitter *hashtags*, and particularly hashtag recommendation, we have identified the recent study presented in [22], where an approach for hashtag recommendation is introduced. This approach computes a similarity measure between *tweets* and uses a ranking system to recommend *hashtags* to new *tweets*. A different approach is proposed in [23], where an event detection method is

described to cluster Twitter *hashtags* based on semantic similarities between the *hashtags*. Two methods for *tweet* vector generation are proposed and their performance evaluated on clustering and event detection in comparison to word-based vector generation methods. This work is in line with our work except for the fact that the semantic similarities are computed based on the message content similarities rather than being based on semantic hashtag similarities.

The foundation of our proposal is the use of meta-classes to boost the performance of Twitter messages. Although the application on a Twitter classification problem is a novel contribution, the use of meta-classes has been studied in other classification contexts. In [24] the use of meta-classes is proposed to improve the performance of classifiers in an handwritten character recognition problem. The use of meta-classes is promoted in this study based on the complex boundaries between classes, the classes overlapping and the lack of sufficient number of samples for some classes.

The related work presented so far sheds light on the importance of social networks in the scientific community, specially the recently explored niche of Twitter *hashtags*, that can have multiple applications like recommendation systems or improvement of search capabilities. In the next section we will detail our proposed approach in order to settle our contribution in the field.

3 Proposed Approach

This section describes the proposed approach to define meta-hashtags and to use them in a classification application to improve the overall classification obtained. Our approach is twofold, resulting in two final models, the baseline model, that considers the user-defined *hashtags*, and the meta-hashtags model, that considers the clusters of similar messages grouped by a single meta-hashtag. In Fig. 1 we depict the proposed framework.

The baseline model is constructed and trained with labelled examples that use the user-defined *hashtags* as a "one-against-all" two-class problem. In the meta-hashtags model, semantic meta-hashtags were heuristically defined, by clustering similar in a meta class. Related classes are then relabelled according to the new defined meta-hashtags and a similar training process occur in order to construct the new proposed model.

The underpinning idea behind the use of meta-hashtags is to combine the class label of similar messages in order to mitigate the effects of the bias introduced by freely user-defined *hashtags*, and thus improving the overall classification of Twitter messages according to their *hashtags*.

4 Experimental Setup

In this section we start by describing the built data set for the purpose of testing our approach. We also characterize the methodology for documents representation. We then proceed dealing with the pre-processing method and finally, we conclude by introducing the performance metrics used to evaluate the proposed approach.

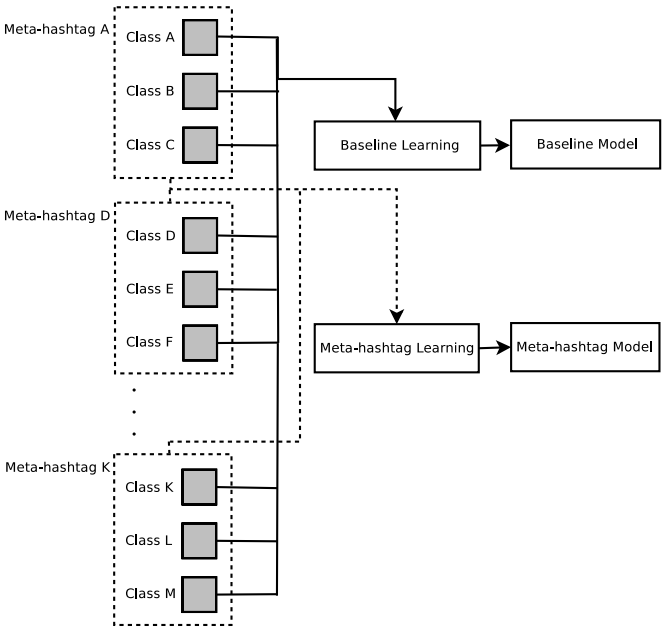


Fig. 1. Proposed approach

4.1 Dataset

The dataset was constructed by requesting public *tweets* to the Twitter API[1]. We have collected more than 230.000 messages during four days and, considering the worldwide usage of Twitter, *tweets* were only considered if the user language was defined as English. All the messages that did not have at least one hashtag were discarded, as the hashtag are assumed as the message classification. Finally, *tweets* containing no message content besides *hashtags* were also discarded and all the *hashtags* are removed from remaining *tweets*. From the 230.000 collected messages, we reach 10.000 *tweets* that have a body part and at least one hashtag.

As users are able to define their own *hashtags*, a high number of different *hashtags* is present in the requested *tweets*. In order to narrow the number of classes, we have only considered the most used *hashtags*. Finally, we used the crowdsourcing platform *http://tagdef.com/* to discover the hashtag meaning and the related *hashtags*, and considered the most used ones, which have at least two or more related *hashtags*, so a meta-hashtag class may empirically be defined. A total number of 15 *hashtags* were found to match this presumption and a total number of 1.230 *tweets* were considered as being labelled and suited for classification purposes. The individual *hashtags* were semantically clustered in 5 meta-hashtag classes, as depicted in Table 1.

[1] `https://dev.Twitter.com/`

Table 1. Amount of positive documents in the training and testing phases

	Training	Testing
NP	138	139
NOWPLAYING	72	68
meta-hashtag NP	**209**	**207**
SEX	70	54
PORN	65	56
XXX	23	19
HOT	14	6
meta-hashtag SEX	**104**	**76**
JOB	32	36
JOBS	41	37
meta-hashtag JOB	**59**	**61**
NW	32	32
NOWWATCHING	7	4
meta-hashtag NW	**39**	**36**
TEAMFOLLOW	17	18
TEAMFOLLOWBACK	126	148
FOLLOWBACK	14	24
NF	58	57
NOWFOLLOWING	7	10
meta-hashtag NF	**207**	**238**

The *tweets* were then split into two equally sized and disjoint sets: training and testing. The training data set is used to build classification learning models, and the testing data set to evaluate performance.

Table 1 describes the positive documents of each class and the corresponding meta-class of the data set. As can be observed from the Table 1, there is an heterogeneous distribution of *hashtags* in the dataset. For example, class TEAMFOLLOWBACK has 274 documents, while class NOWFOLLOWING has only 17 documents. The amount of positive documents in the training and testing data sets is balanced because it is obtained by the equally split of training and test sets.

4.2 Pre-processing Methods

A *tweet* is represented as one of the most successful and commonly used document representation, which is the vector space model, also known as *Bag of*

Words. The collection of features is built as the dictionary of unique terms present in the documents collections. Each document of the document collection is indexed with the *bag* of the terms occurring in it, i.e., a vector with one element for each term occurring in the whole collection.

High dimensional space can cause computational problems in text-classification problems where a vector with one element for each occurring term in the whole connection is used to represent a document. Also, overfitting can easily occur which can prevent the classifier to generalize and thus the prediction ability becomes poor. In order to reduce feature space pre-processing methods are often applied. These techniques aim at reducing the size of the document representation and prevent the mislead classification as some words, such as articles, prepositions and conjunctions, called *stopwords*, are non-informative words, and occur more frequently than informative ones. These words could also mislead correlations between documents so *stopword* removal technique was applied. *Stemming* method was also applied. This method consists in removing case and inflection information of a word, reducing it to the word stem. Stemming does not alter significantly the information included, but it does avoid feature expansion.

4.3 Learning and Evaluation

The evaluation of our approach was done by the dataset with the Support Vector Machine (SVM) method. This machine learning method was introduced by Vapnik [25], based on his Statistical Learning Theory and Structural Risk Minimization Principle. The idea behind the use of SVM for classification consists on finding the optimal separating hyperplane between the positive and negative examples. Once this hyperplane is found, new examples can be classified simply by determining which side of the hyperplane they are on. SVM constitute currently the best of breed kernel-based technique, exhibiting state-of-the-art performance in text classification problems [26–28]. SVM were used in our experiments to construct the model with user-defined *hashtags* and the meta-hastags model.

In order to evaluate the binary decision task of the proposed models we defined several measures based on the possible outcomes of the classification, such as, error rate ($\frac{FP+FN}{TP+FP+TN+FN}$), recall ($R = \frac{TP}{TP+FN}$), and precision ($P = \frac{TP}{TP+FP}$), as well as combined measures, such as, the van Rijsbergen F_β measure [29], which combines recall and precision in a single score.

F_β is one of the best suited measures for text classification used with $\beta = 1$, i.e. F_1 ($F_1 = \frac{2*P*R}{P+R}$), an harmonic average between precision and recall.

5 Experimental Results and Analysis

In this Section we evaluate the performance obtained on the Twitter data set using the two approaches described in Section 3, namely the baseline approach considering the 15 initial *hashtags* and the meta-hashtag approach. Table 2 summarises the performance results obtained by classifying the datasets.

Table 2. Comparative results

	Baseline Model			Meta-hashtags Model		
	Precision	Recall	F1	Precision	Recall	F1
NP	45.73%	53.96%	49.50%	35.44%	92.81%	51.29%
NOWPLAYING	23.96%	33.82%	28.05%	15.11%	80.88%	25.46%
meta-hashtag NP				**50.55%**	**88.89%**	**64.45%**
SEX	70.18%	74.07%	72.07%	38.69%	98.15%	55.50%
PORN	80.00%	71.43%	75.47%	40.88%	100.00%	58.03%
XXX	21.05%	21.05%	21.05%	11.45%	100.00%	20.54%
HOT	7.14%	16.67%	10.00%	3.61%	100.00%	6.98%
meta-hashtag SEX				**69.23%**	**94.74%**	**80.00%**
JOB	53.33%	66.67%	59.26%	34.31%	97.22%	50.72%
JOBS	50.79%	86.49%	64.00%	33.33%	91.89%	48.92%
meta-hashtag JOB				**56.86%**	**95.08%**	**71.17%**
NW	17.65%	9.38%	12.24%	15.38%	12.50%	13.79%
NOWWATCHING	0.00%	0.00%	-	3.85%	25.00%	6.67%
meta-hashtag NW				**19.23%**	**13.89%**	**16.13%**
TEAMFOLLOW	100.00%	100.00%	100.00%	26.09%	100.00%	41.38%
TEAMFOLLOWBACK	40.80%	55.41%	46.99%	32.91%	87.84%	47.88%
FOLLOWBACK	0.00%	0.00%	-	5.57%	91.67%	10.50%
NF	25.00%	5.26%	8.70%	13.92%	96.49%	24.34%
NOWFOLLOWING	-	0.00%	-	2.53%	100.00%	4.94%
meta-hashtag NF				**54.94%**	**91.18%**	**68.56%**

Analysing the table we can observe that the use of a meta-hashtag outperforms the overall classification of the initial *hashtags* when they are considered individually. For example, the class NP has a F1 measure of 49.50% in the baseline model, in the class NOWPLAYING the corresponding value is 28.05% and with the use of a meta-hashtag the new proposed model presents a F1 value for the meta-hashtag NP class of 64.45%. This might be related to the fact that the content being classified in the meta-hashtag dataset is different from the initial dataset, thus misleading the classifier. Theses improvements on the results obtained may also be observed for other cases, like in class JOB with F1 of 59.26% and class JOBS with 64.00%, while the corresponding meta-hashtag JOB has 71.17%.

With the use of a meta-hashtag we unify the labelling process by grouping similar messages and placing them in the same classification class, thus boosting the performance of the overall classifier. This analysis is in line with previous

results in [23] where from a set of different methods proposed, a pseudo-meta-hashtag approach was presented to be beneficial in a clustering problem.

Other noteworthy results are obtained when using the meta-hashtag model to classify the initial classes. Although the F1 measure decreases considering the baseline model, the recall increases in a higher proportion. As an example, the class XXX classified by the baseline model presents a precision and a recall of 21.05%. Classified by the meta-hashtag model, the precision falls to 11.45% and the recall raises up to 100.00%, which means that precision proportionally decreases less than the increase of recall. This is due to the false positive increase being less than the increase of true positives. The increase of false positives, and thus the decrease of the F1 measure was expected, as we mislead the classifier by training it with the meta-hashtags examples, which means we used as positive examples not only the initial class messages, but also the related messages that belong to semantically similar *hashtags*.

In the baseline model, classes like NOWFOLLOWING or FOLLOWBACK have no F1 measure. This occurs because the classifier did not identified any true positive document, probably due to the lack of information in the training phase, so precision and recall are 0.00% and F1 can not be calculated. In these classes the use of the meta-hashtags approach, more than increasing the classifier performance, permits the identification of these classes documents.

6 Conclusions and Future Work

In this paper we have presented a meta-hashtag approach in order to deal with the bias produced by freely user-defined hashtag in Twitter social network. The main idea of defining semantic meta-hashtags that cluster similar messages is to boost the classification performance of messages, by avoiding the mislead classification problem raised by having multiple classes to identical features.

For that purpose, we have constructed a dataset by requesting public *tweets* to the Twitter API and conducted a set of experiments by comparing a baseline model based on user-defined *hashtags* with our approach based on meta-hashtags.

The preliminary results are very promising. It is possible to observe that the proposed approach outperforms the F1 measure of each initial class included in its composition, with the exception of the initial class TEAMFOLLOW that is already correctly classified in the initial approach. It is also important to note the overall improvement of the recall metric when using the meta-hashtag model to classify the initial classes. This improvement sustains the use of meta-hashtags and makes it possible to infer that, the more information we give to the classifier in the training process, the better for the identification of true positive documents in the testing phase, as the increase of recall is due to the increase of true positives.

Our future work will give a formalization of the meta-hastags model including *hashtags* similarities in order to evaluate the clustering quality. Moreover, evaluation of dynamically created Twitter meta-hashtags will be performed possibly for enrichment of applications in real-time.

Acknowledgment. This work is financed by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «PEst-C/EEI/LA0014/2011»

References

1. Efron, M.: Hashtag retrieval in a microblogging environment. In: SIGIR, pp. 787–788 (2010) 226
2. Abel, F., Gao, Q., Houben, G.-J., Tao, K.: Semantic enrichment of twitter posts for user profile construction on the social web. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part II. LNCS, vol. 6644, pp. 375–389. Springer, Heidelberg (2011) 226
3. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: Proceedings of the 19th International Conference on World Wide Web, pp. 851–860 (2010) 226, 227
4. Popescu, A.-M., Pennacchiotti, M.: Detecting controversial events from twitter. In: CIKM, pp. 1873–1876 (2010) 226, 227
5. Becker, H., Naaman, M., Gravano, L.: Beyond trending topics: Real-world event identification on twitter. In: ICWSM (2011) 226
6. Ovadia, S.: Exploring the Potential of Twitter as a Research Tool. Behavioral & Social Sciences Librarian 28(4), 202–205 (2009) 226
7. Rowe, M., Stankovic, M.: Mapping tweets to conference talks: A goldmine for semantics. In: Workshop on Social Data on the Web, SDoW (2010) 226
8. Weller, K., Puschmann, C.: Twitter for Scientific Communication: How Can Citations/References be Identified and Measured?, pp. 1–4 (2011), `http://journal.webscience.org/500/` 226
9. Abel, F., Gao, Q., Houben, G.-J., Tao, K.: Analyzing user modeling on twitter for personalized news recommendations. In: Proceedings of the 19th International Conference on User Modeling, Adaption, and Personalization, pp. 1–12 (2011) 226
10. Tumasjan, A., Sprenger, T.O., Sandner, P.G., Welpe, I.M.: Predicting elections with twitter: What 140 characters reveal about political sentiment. In: Proceedings of the Fourth International AAAI Conference on Weblogs and Social Media, pp. 178–185 (2010) 226
11. O'Connor, B., Balasubramanyan, R., Routledge, B.R., Smith, N.A.: From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In: Proceedings of the International AAAI Conference on Weblogs and Social Media (2010) 226, 227
12. Huang, J., Thornton, K.M., Efthimiadis, E.N.: Conversational tagging in twitter. In: Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, pp. 173–178 (2010) 227
13. Merriam-webster's dictionary (October 2012), `www.merriam-webster.com/` 227
14. Zappavigna, M.: Ambient affiliation: A linguistic perspective on Twitter. New Media & Society 13(5), 788–806 (2011) 227
15. Johnson, S.: How twitter will change the way we live. Time Magazine 173, 23–32 (2009) 227

16. Tsur, O., Rappoport, A.: "What's in a hashtag?: content based prediction of the spread of ideas in microblogging communities". In: Proceedings of the Fifth ACM International Conference on Web Search and Data Mining, WSDM 2012, pp. 643–652 (2012) 227
17. Yang, L., Sun, T., Zhang, M., Mei, Q.: We know what @you #tag: does the dual role affect hashtag adoption? In: Proceedings of the 21st International Conference on World Wide Web, WWW 2012, pp. 261–270 (2012) 227
18. Chang, H.-C.: A new perspective on twitter hashtag use: diffusion of innovation theory. In: Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem, ASIS&T 2010, vol. 47, pp. 85:1–85:4 (2010) 227
19. Doerr, B., Fouz, M., Friedrich, T.: Why rumors spread so quickly in social networks. Commun. ACM 55(6), 70–75 (2012) 227
20. Tantipathananandh, C., Berger-Wolf, T.Y.: Finding Communities in Dynamic Social Networks. In: 2011 IEEE 11th International Conference on Data Mining, pp. 1236–1241 (2011) 227
21. Treiber, M., Schall, D., Dustdar, S., Scherling, C.: Tweetflows: flexible workflows with twitter. In: Proceedings of the 3rd International Workshop on Principles of Engineering Service-Oriented Systems, pp. 1–7 (2011) 227
22. Zangerle, E., Gassler, W., Specht, G.: Recommending #-tags in twitter. In: Proceedings of the Workshop on Semantic Adaptive Social Web, in Connection with the 19th International Conference on User Modeling, Adaptation and Personalization, UMAP 2011, pp. 67-78 (2011) 227
23. Oguztuzun, H., Ozdikis, O., Senkul, P.: Semantic Expansion of Hashtags for Enhanced Event Detection in Twitter. In: VLDB The First International Workshop on Online Social Systems (2012) 227, 233
24. Koerich, A.: Improving classification performance using metaclasses. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 717–722 (2003) 228
25. Vapnik, V.: The Nature of Statistical Learning Theory. Springer (1999) 231
26. Joachims, T.: Learning Text Classifiers with Support Vector Machines. Kluwer Academic Publishers, Dordrecht (2002) 231
27. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. The Journal of Machine Learning Research 2, 45–66 (2002) 231
28. Costa, J., Silva, C., Antunes, M., Ribeiro, B.: On using crowdsourcing and active learning to improve classification performance. In: Proceedings of 11th International Conference on Intelligent Systems Design and Applications (ISDA), pp. 469–474 (2011) 231
29. van Rijsbergen, C.: Information Retrieval. Butterworths (1979) 231

Reinforcement Learning and Genetic Regulatory Network Reconstruction

Branko Šter and Andrej Dobnikar

Faculty of Computer and Information Science, University of Ljubljana
Tržaška 25, 1000 Ljubljana, Slovenia
branko.ster@fri.uni-lj.si

Abstract. Many different models of genetic regulatory networks (GRN) exist, but most of them are focused on off-line processing, so that important features of real networks, like adaptive and non-stationary character are missed. Interdisciplinary insight into the area of self-organization within the living organisms has caused some interesting new thoughts, and the suggested model is among them. Based on reinforcement learning of the Boolean network with random initial structure, the model is searching for a specialized network, that agrees with experimentally obtained data from the real GRN. With some experiments of real biological networks we investigate its behaviour.

Keywords: genetic regulatory network, reinforcement learning, canalysing Boolean functions.

1 Introduction

Understanding the way biological cells process an enormous number of operations, required for normal activity, is a major challenge in developmental biology. A number of different approaches toward modelling a genetic regulatory network (GRN) have been introduced, including linear models [1], Bayesian networks [2], neural networks [3] and neurogenetic approaches [19], differential equations [4], and models including stochastic components on the molecular level [5]. A good review of general models is given in [6]. The Boolean network (BN) model was originally introduced by Kauffman [7], and has recently been used by many authors (to mention only some of them [9, 10]). A stochastic nature of GRN was reconstructed either with context-sensitive probabilistic Boolean networks together with a dynamic programming approach [11] or with a conditional mutual information [12]. In one of our recent works we studied the origin of a subcellular signal transduction system via an evolutionary (genetic) algorithm [20], which might be called a bottom-up approach because of its growth feature. In this paper, we however change the direction and suggest top-down access to the GRN from an evolved structure (only in terms of number of nodes or genes/proteins and number of inputs to nodes) to the specialized network of GRN. Though many different models of the GRN obviously exist, they do not explain the on-line operation of the subcellular structures of genes and/or proteins.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 236–245, 2013.

They are mostly able to process the data, that are experimentally obtained in some individual instances more or less successfully. The self-organizing feature or capability of adaptation is therefore missing. We believe that with the reinforcement learning algorithm we are able to explain some of the details that are not included in the current models. In particular, the proposed model enables us to deal with the time-varying structure of genes/proteins in order to ensure functional continuity.

In the paper we introduce an adaptive Boolean model of GRN (ABGRN) with a reinforcement learning algorithm of special regulatory signals that follow the measured data from a biological cell. The search for suitable function pathways in the universal and randomly constructed network is the goal of the algorithm. The following characteristics are presumed: 1. The model represents an adaptive and non-stationary network of gene/protein nodes with special canalysing Boolean functions; 2. Intercell or intracell (transcription) signals act as inputs to the ABGRN, though the model can work without inputs; 3. Genes and proteins (mRNAs) determine states and outputs of ABGRN, respectively; 4. Instead of bio-chemical mechanisms in real systems, measured data enable assessment of the states and/or outputs in ABGRN - the response is either reward (R) that preserves the network or penalty (P) that causes adequate updating of the links and/or function nodes in the network; 5. Special regulatory signals adjust connections between nodes and their functions to minimize the penalties; 6. There are some global (evolving) parameters, n (number of nodes), k (number of inputs to nodes) and m (number of regulatory signals), whose correct values represent the **necessary condition** for efficient ABGRN, while connections between nodes and their functions establish a **sufficient condition**; parameter d (number of cell-differentiating signals) implies cell types (differentiation) in a similar way to m parameter, and will therefore not be considered in this work.

The goal of the presented model is to recognize the principle of cell processing of the sub-cell network of genes/proteins based on an adaptive process of self-organisation. The reinforcement learning of regulatory signals is introduced to raise the proper function pathways in the network so that the penalties related to the measured data from real GRN are minimized.

The paper is organized as follows. In section 2, basic definitions of the suggested ABGRN model are given, followed by the description of the reinforcement learning algorithm in section 3. The experimental work is described in section 4, together with the results and comments. In the conclusion we resume the main issues and give some ideas for the future work.

2 Basic Definitions

Adaptive Boolean network (ABN) is an expansion of the Boolean network (BN), where BN = (V, B). V is a set of nodes (genes or proteins), $V = (x_1, \ldots, x_n)$, and B a set of Boolean functions, $B = (f_1, \ldots, f_n)$, with $f_i = f_i(x_{i1},\ldots x_{ik})$, where ik is the number of inputs to node i, and ABN = (V, R, B^*), where V is again the set of nodes, R is a set of regulatory signals representing two groups, $R = (r_1,\ldots, r_i,\ldots, r_p, r_{p+1}, \ldots, r_j,\ldots, r_{p+q})$, first p influencing connections in the network and next q establishing functions of the nodes. B^* is a special set of canalysing Boolean functions, $B^* = (f_1^*, \ldots, f_{2x2^k}^*)$. There

are $m = p + q$ regulatory signals, but only one at a time is used to change the regulatory point. The selection of a regulatory signal from R depends on the probabilities p' for the signals within the first group and q' for signals from the second group. In the experimental work we use $p' = 0.1$ and $q' = 0.9$ to improve the convergence speed of the learning. Namely, any change of connection between two nodes needs more functional updates on the sequel pathways in order to properly follow the data from the GRN via experimental work. Connection between two nodes in the network can either be opened ($r_i = 0$) or closed ($r_i = 1$), $i = 1,\ldots, p$, while node function r_j can be defined from set B^*, which defines one of the 2^*2^k special canalysing Boolean functions. Each $f_j^* = r_j$, has only one active value (T or F) and 2^k-1 non-active values (F or T), where k is the number of j-th node inputs. In such functions, any input variable (so called canalysing input) is able to establish the output either as a constant or a type of gate function (AND/NAND) of the other k - 1 inputs of all possible combinations of their literals. Instead of the AND/NAND pair one can deal with the OR/NOR pair which is related with the first pair through the DeMorgan theorem. There are other canalysing functions as well, but we restrict ourselves to this set in order to reduce the set and therefore simplify the learning procedure.

The network with random connections and random distribution of special canalysing Boolean functions represents a universal pool for the processing of any cell function. In order to specialize for a specific task, the GRN needs to adapt through some bio-chemical mechanisms, enabling self-organization. In living organisms there exist many so called switch proteins controlled by phosphorylation, organised in cascades, transmitting the signal onward and, in the process, amplifying, distributing, and modulating it, making such mechanisms feasible [13]. The process of self-organisation is vital for universal network adaptation to a specialized network of a cell task. As the network is non-stationary, adaptation is a permanent duty of the cell regulatory mechanism. In the ABGRN model, reinforcement learning of regulatory signals is considered to perform such specialization. Each element of R, r_l, $l = 1, \ldots, m$, is bound to a randomly selected connection (between two nodes) or function node and is an object of reinforcement learning. It is assigned to change the value of the point (type of connection or function of node) in a one at a time manner. The proper values of parameters n (number of nodes that defines the size of network), and m (number of control points or signals) of ABGRN are essential for the successful learning process. The parameter k, representing the number of inputs to the nodes, was shown to be rather small in real biological systems [8, 13], which limits the frame of our simulations to the values of k between 2 and 5.

3 Reinforcement Learning in ABGRN

The adaptive Boolean model of the gene regulatory network (ABGRN) is a dynamic system which is changing its states (values of genes) in a synchronous way. It was shown [21, 22] that artificial synchrony simplifies computation while preserving the qualitative, generic properties of global network dynamics. The transition of states is caused either by change(s) of input signal(s) and/or gene state(s) in a particular instant. In the model, all nodes update simultaneously, according to the functions. The system is following a given trajectory of the global states (states of all nodes). All global states can be divided into three groups: the group of states that belong to the so

called states of 'Gardens-of-Eden', the transient states and the attractor states. The second are temporary states as they are passed towards the limiting cycle of attractor state(s). The limiting cycle represents a steady state of one or more attractor states. In the former case, the steady state of the network is unique, while in the latter the steady state is defined with the cycle of attractor states. The response of the model network is measured with the percentage of matching the next states with that of the data (from real GRN), averaged over C possible starting states. In order to assess the learning of the model, the comparison with state transitions of a real system is crucial. These data are obtained with micro array technology (many genes from a few biological replicates) or flow cytometry (few genes from many biological samples or replicates) [13]. There are obviously two factors that influence the successful state transitions of the model. One is the structure (size and connectivity) of the model, defined with parameters n, k and regulatory signals r_i, $i = 1, \ldots, p$, and the other the functions of the nodes, defined by their types and regulatory signals r_j, $j = 1, \ldots, q$. Through reinforcement learning, the connectivity and the function nodes are affected, which makes an expectation for finding a correct structure that is compatible with the measured data (or real GRN) realistic.

The hypothesis for the self-organisation in ABGRN is the following: given the measured data from the real GRN that imply a node connectivity graph with logic (next state) equations of related nodes (genes) and initial (random) model network with relevant parameters (n, k, m), it is possible to reconstruct GRN for the cell task by applying the reinforcement learning algorithm. The choice of relevant parameters is of the essence; however, efficiency of the learning from the measured data is the main achievement of the approach.

The reinforcement learning algorithm can be described as follows. The initial ABGRN is raised with random connections (structure) of random functional nodes. The regulatory signals are bound to m random regulatory points, defining p connections and q functions of the nodes, where p and q are important parameters of the algorithm and should be congruent with the number expected to be changed in real systems during some period (M iteration steps), and small because of their impact on the time of adaptation. The proper values of the regulatory signals are objects of the reinforcement learning and influence the effectiveness of the model. The algorithm starts with a set of C (possibly all) global states, and initial random regulatory signals. The next states of the model are processed and the difference from the target data (from real GRN) is recognized, resulting in a measure of matching.

A regulatory signal is selected from R. It may be a changed connection (with smaller probability like $p' = 0.1$) or a changed function (with probability like $q' = 0.9$). In case of penalty, the last change of the regulatory signal is suppressed. In case the result of the change is reward (improved matching), the applied regulatory signal is accepted and its efficiency updated. In either case the procedure continues with selection and modification of another regulatory signal and the same set of states is initiated again, followed by the next states processing to measure the matching.

After M iterations of C transitions, m new regulatory points are selected by considering the result from the hitherto learning implicitly included in current efficiencies of the regulatory points (connections and functions). Greater efficiency means higher probability for selection. The efficiency of a regulatory signal is measured during the run of the algorithm. The regulatory signal influencing connection considers the

percentage of the successful connections and proximity (distance to a gene in terms of number of nodes). In case of a regulatory signal for the function node, it reflects the node proximity to a relevant gene only and is higher the shorter the distance to a gene. A new connection is established when the following criterion is satisfied:

$$random(0,1) < eff(new\_con) * \left(1 - eff(old\_con)\right) \,\&\, random(0,1) < \frac{1}{prox + 1}$$

where *eff* is efficiency, *prox* is proximity or a distance of an observed point to a gene, *new_con* and *old_con* are new and old regulatory connections. In this way a more efficient connection has a higher probability to be selected than a less efficient connection, and a node closer to the gene is more probable to be chosen as a new regulatory point.

The result of the learning is given with average percentage of the matching between the next states of the model and the data. The algorithm stops after a predefined number of repetitions. Figure 1 shows a simplified pseudo-code of the reinforcement learning algorithm.

```
begin
  create initial ABGRN with global parameters n, k, m
  define initial p and q regulatory signals
  while (no stopping condition)
    begin
      select r1 with p', q', considering also efficiency
      select C starting global states
      process model
      compare next global states with data
      assess matching
      if Penalty (worse matching than before)
        restore signal
      if Reward
        update efficiency of regulatory signal r1
      after every M iterations define new p and q signals
    end;
end.
```

Fig. 1. Pseudo-code of reinforcement learning algorithm (see text for further explanation)

4 Experimental Work

We applied the suggested approach on two examples, found in [15], that represent the attempts to model real biological networks.

The first example deals with coupled oscillations in the cell cycle. It was apparent that the onset of M (mitosis) and S (DNA replication) phases of the cell cycle are controlled by the periodic activation of cyclin-dependent kinases [14]. From the differential equations model of the system, the logic and the Boolean model were derived [15], and the resulting graph is shown in Fig. 2. The nodes represent genes while their state transition functions depict the need for the proteins/genes with appropriate canalysing functions.

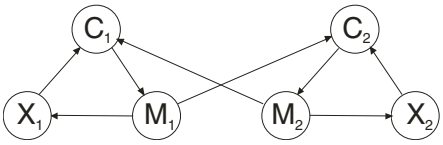


Fig. 2. The directed graph of coupled oscillations in cell cycle

The logic equations for the nodes (genes) in the graph are (↓ is NOR):

$C_1(t+1) = X_1(t) \downarrow M_2(t)$ $\quad C_2(t+1) = X_2(t) \downarrow M_1(t)$
$X_1(t+1) = M_1(t)$ $\quad X_2(t+1) = M_2(t)$
$M_1(t+1) = C_1(t)$ $\quad M_2(t+1) = C_2(t)$

Several simulation runs were carried out with different global parameters n and m. The goal was to find the most efficient specialized network in terms of time and accuracy obtained with the reinforcement learning from the random starting network, with the smallest number of nodes and regulatory signals. For every combination of global parameters, three runs of 100,000 iterations were accomplished and their average is shown in the resulting diagrams, depicted in Fig. 3, where p^* (q^*) is proportion of connections (functions) in R, relative to all connections (nodes) in the network. Parameter M was set to 10,000.

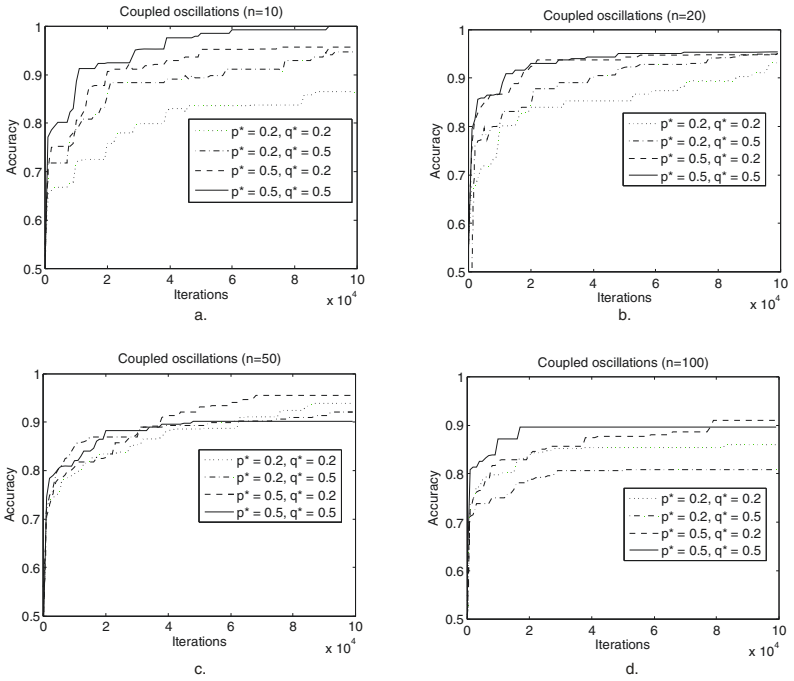


Fig. 3. The resulting diagrams of simulation runs for Example 1: a) n = 10, b) n = 20, c) n = 50, and d) n = 100

To illustrate the result of reinforcement learning, one of the combinations of efficiency and global parameters is demonstrated with the Pajek tool [16] in Fig. 4. Square nodes show genes and circular nodes represent genes/proteins that are responsible for state transition functions. This is why the number of nodes in the model is higher than the number of genes. With black colour and solid lines for function nodes and connections, the results of learning are emphasized.

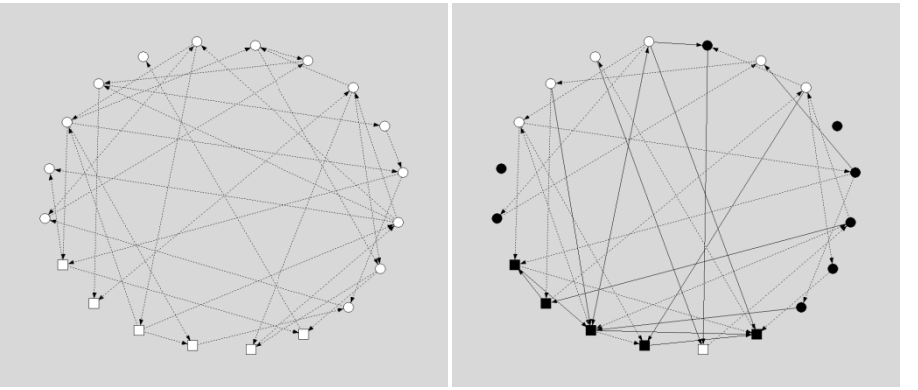


Fig. 4. Illustration of initial and resulting networks in case of Example 1, $n = 20$, $p^* = q^* = 0.2$. Nodes with changed functions are marked black and new connections are drawn with solid lines.

In the second example the Boolean model of cell growth, differentiation, and apoptosis (programmed cell death) is applied, originally introduced by Huang and Ingber [17]. The connectivity graph is shown in Fig. 5, where again the nodes represent genes and their transition functions show the demand for the proper proteins/genes functions.

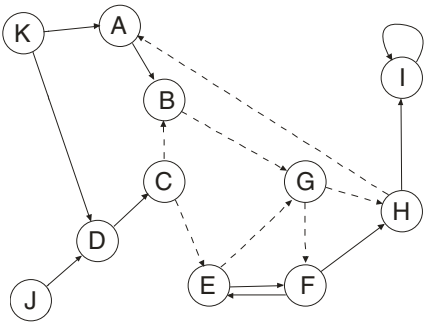


Fig. 5. The connectivity graph of cell growth, differentiation, and apoptosis: solid lines indicate positive and dotted lines negative interactions

This time the logic functions are (| is NAND, · is AND, ' is negation):

A(t+1) = K(t)·H'(t)	B(t+1) = A(t)·C'(t)	
C(t+1) = D'(t) + I(t)	D(t+1) = J(t)·K(t)	
E(t+1) = C'(t) + F(t)	F(t+1) = E'(t) + G(t)	
G(t+1) = B(t) | E(t)	H(t+1) = F(t)·G'(t)	
I(t+1) = H(t)·I'(t)	J(t+1) = J(t)	K(t+1) = K(t)

Assuming that the growth factor (node K) and cell spreading (node J) are both True (1), and due to implication J = K = 1 → D = 1 (see equation for D), a nontrivial growth attractor exists [18], which enables one to use only 8 nodes for the initial global states and for the sequencing of the system.

Simulation runs were performed in a similar way to the previous example. The resulting diagrams are depicted in Fig. 6.

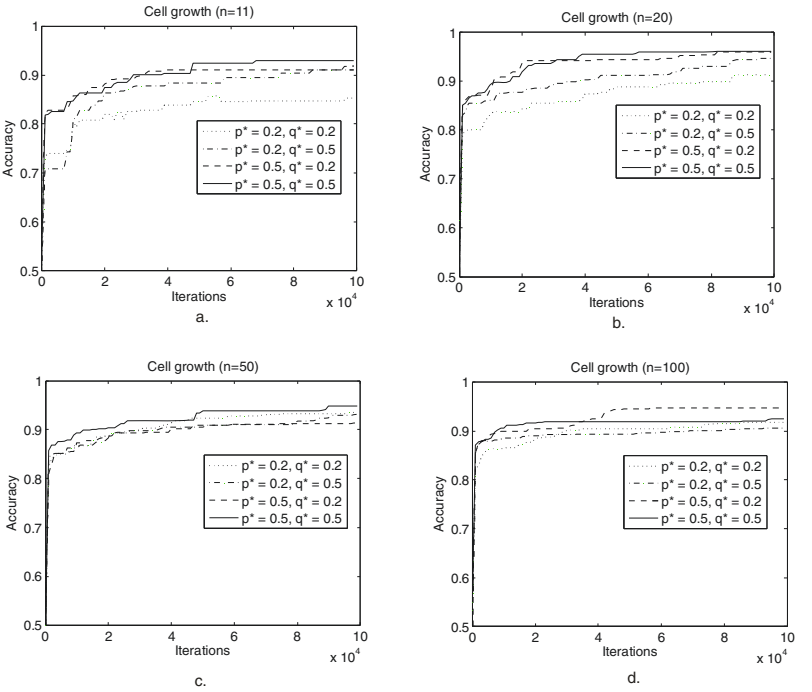


Fig. 6. The resulting diagrams of simulation runs for Example 2: a) $n = 11$, b) $n = 20$, c) $n = 50$, and d) $n = 100$

Some comments are in order. It is evident from Fig. 3, that for Example 1 (coupled oscillations), network with $n = 10$ yields near 100% accuracy at $p^* = q^* = 0.5$, which is too robust or time consuming to be optimal. With only one partition equal to 0.2 we already get the acceptable result in terms of accuracy and number of iterations. But with the network of 20 or more nodes, for any partitions of p^* and q^*, the same

accuracy of near 95% is obviously obtained much later. Larger networks are therefore too complex for the coupled oscillation task. The simulation runs of Example 2 (cell growth) shown in Fig. 6, clearly indicate that a network with 11 nodes (which is the same as the number of genes) is not big enough and therefore needs more protein (function) nodes for better accuracy. The network with 20 nodes and p^* or $q^* = 0.2$ seems to be optimal as it achieves around 95% accuracy. Other results are worse again in terms of time/performance trade off.

Common to both experiments is the observation that with the increased complexity of a cell task, the size of the required network and the number of regulatory signals are increased. This is in concordance with the natural expectation, that a more complex network with a higher number of possible topologies or functional pathways implies a harder combinatorial problem, which requires more decisions to be taken.

5 Conclusion

A new model of genetic regulatory network is introduced. It is based on the Boolean network with restricted set of canalysing functions and regulatory signals that enable modification of initial random network. The reinforcement learning is able to search for the specialized network that matches with the real GRN and its experimentally obtained data. It represents an attempt to describe the biological self-organization of the GRN of genes and/or proteins from the origin of the living system. The real biological examples were used to prove the efficiency of the model. We believe that the results are interesting as they show a new paradigm for further discoveries within the sub-cell world. There are many unknown details yet to be discovered, mostly in biochemical mechanisms and gene proximity functions that are supposed to perform self-organization, to mention only some of them. There remains plenty of room for research, indeed.

References

1. D'Haeseleer, P., Wen, X., Fuhrman, S., Somogyi, R.: Linear modeling of mRNA expression levels during CNS development and injury. In: Pac. Symp. Biocomp., vol. 4, pp. 41–52 (1999)
2. Murphy, K., Mian, S.: Modeling gene expression data using dynamic Bayesian networks. Techn. Report, Berkeley (1999)
3. Weaver, D.C., Workman, C.T., Stormo, G.D.: Modeling regulatory networks with weight matrices. In: Pac. Symp. Biocomp., vol. 3, pp. 89–102 (1999)
4. Chen, T., He, H.L., Church, G.M.: Modeling gene expression with differential equations. In: Pac. Symp. Biocomp., vol. 4, pp. 29–40 (1999)
5. Arkin, A., Ross, J., McAdams, H.H.: Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli. Genetics 149(4), 1633–1648 (1998)
6. de Jong, H.: Modeling and simulation of genetic regulatory systems: a literature review. J. Comput. Biol. 9, 69–103 (2002)
7. Kauffman, S.A.: Metabolic stability and epigenesis in rndomly constructed genetic nets. J. Theoret. Biol. 22, 437–467 (1969)

8. Kauffman, S.A.: The Origins of Order. Oxford Univ. Press (1993)
9. Akutsu, T., Miyano, S., Kuhara, S.: Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics 16, 727–734 (2000)
10. Shmulevich, I., Dougherty, E., Wei, Z.: From Boolean Probabilistic Boolen Networks as Models of Genetic Regulatory Networks. Proc. of IEEE 90(11) (2002)
11. Pal, R., Datta, A., Bittner, M., Dougherty, E.R.: Intervention in context-sensitive probabilistic Boolean networks. Bioinformatics 21, 1211–1218 (2005)
12. Liang, K.C., Wang, X.: Gene Regulatory Network Reconstruction Using Conditional Mutual Information. Journal of Bioinformatics and Systems Biology (2008)
13. Alberts, B., Bray, D., Hopkin, K., Johnson, A., Lewis, J., Raff, M., Roberts, K., Walter, P.: Essential Cell Biology, 3rd edn. Garland Science (2010)
14. Stillman, B.: Cell cycle control of DNA replication. Science 274, 1659–1664 (1996)
15. Heidel, J., Maloney, J., Farrow, C.: Finding cycles in synchronous Boolen networks with applications to biochemical sytems. Int. Journal of Bifurcation and Chaos 13, 535–552 (2003)
16. Batagelj, V., Mrvar, A.: Pajek-program for large network analysis. Connections 21, 47–57 (1998)
17. Huang, S., Ingber, I.: Shape-dependent control of cell growth, differentiation, and apoptosis: switching between attractors in cell regulatory networks. Exp. Cell Res. 261, 91–1103 (2000)
18. Farrow, C., Heidel, J., Maloney, J., Rogers, J.: Scalar Equations for Synchronous Boolean Networks with Biological Applications. IEEE T. on Neural Networks 15(2), 348–354 (2004)
19. Benuskova, L., Kasabov, N.: Modelling brain dynamics using computational neurogenetic approach. Cognitive Neurodynamics 2(4), 319–334 (2008)
20. Šter, B., Avbelj, M., Jerala, R., Dobnikar, A.: On the Origin and Features of an Evolved Boolean Model for Subcellular Signal Transduction Systems. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part II. LNCS, vol. 6594, pp. 383–392. Springer, Heidelberg (2011)
21. Huang, S.: Gene expression profiling, genetic networks, and cellular states: An integrating concept for tumorigenesis and drug discovery. J. Mol. Med. 77, 469–480 (1999)
22. Wuensche, A.: Genomic regulation modeled as a network with basins of attraction. In: Proc. Pacific Symp. Biocomp., vol. 3, pp. 89–102 (1998)

Nonlinear Predictive Control Based on Least Squares Support Vector Machines Hammerstein Models

Maciej Ławryńczuk

Institute of Control and Computation Engineering, Warsaw University of Technology
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland
M.Lawrynczuk@ia.pw.edu.pl

Abstract. This paper shortly describes nonlinear Model Predictive Control (MPC) algorithms for Least Squares Support Vector Machines (LS-SVM) Hammerstein models. The model consists of a nonlinear steady-state part in series with a linear dynamic part. A linear approximation of the model for the current operating point or a linear approximation of the predicted output trajectory along an input trajectory is used for prediction. As a result, all algorithms require solving on-line a quadratic programming problem or a series of such problems, unreliable and computationally demanding nonlinear optimisation is not necessary.

Keywords: Process control, Model Predictive Control, Hammerstein systems, Least Squares Support Vector Machines, soft computing.

1 Introduction

The general idea of Model Predictive Control (MPC) algorithms is to repeatedly use on-line a dynamic model of the process to predict its future behavior and to calculate the future control policy from an optimisation problem in which predicted control errors over a time horizon are taken into account [11,16,17]. Such an approach has a few important advantages over many other control techniques: constraints can be easily imposed on process inputs (manipulated variables) and outputs (controlled variables), MPC algorithms can be efficiently used for multivariable processes and for processes with difficult dynamic properties (e.g. with significant time-delays, the inverse response). In consequence, MPC algorithms have been successfully used for years in numerous advanced applications [15].

The current research is concerned with nonlinear MPC algorithms which use for prediction nonlinear models [17], e.g. neural structures [7,8,10]. Although the feedforward neural network can be efficiently used in practice for modelling of many technological processes, the neural model is entirely a black-box model, i.e. its structure has nothing to do with the technological nature of the process. An interesting and sound alternative is a block-oriented model which consists of separate steady-state and dynamic parts connected in series [5]. It turns out that for many processes the cascade structure is the best choice, since steady-state

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 246–255, 2013.

properties of measurement devices and actuators are frequently significantly nonlinear by nature whereas dynamic properties of the processes are (almost) linear. The Hammerstein model, in which the linear dynamic part follows the nonlinear steady-state part can be efficiently used for modelling, fault detection and control of different technological processes, e.g. chemical reactors [2,13], heat exchangers [1] and distillation columns [1,2].

Support Vector Machines (SVM) can be efficiently used for nonlinear classification and function approximation [12]. An important advantage of SVM is the fact that model identification requires solving convex optimisation problems, typically quadratic programming ones. In Least Squares Support Vector Machines (LS-SVM) the identification procedure is even simpler as only least squares problems are solved [14]. On the contrary, neural networks training requires nonlinear optimisation, in which the problem of local minima is unavoidable.

This paper shortly describes nonlinear MPC algorithms for LS-SVM Hammerstein models (the LS-SVM approximator is used in the nonlinear steady-state part of the model). A linear approximation of the model for the current operating point or a linear approximation of the predicted output trajectory along an input trajectory is used for prediction. As a result, all algorithms are computationally efficient since they require solving on-line a quadratic programming problem or a series of such problems, unreliable and computationally demanding nonlinear optimisation is not necessary. Accuracy and computational efficiency of discussed algorithms are demonstrated for a significantly nonlinear benchmark Hammerstein system.

2 Model Predictive Control Algorithms

In MPC algorithms [11,17] at each consecutive sampling instant k, $k = 0, 1, 2, \ldots$, a set of future control increments

$$\triangle \boldsymbol{u}(k) = [\triangle u(k|k)\ \triangle u(k+1|k) \ldots \triangle u(k+N_{\mathrm{u}}-1|k)]^{\mathrm{T}} \tag{1}$$

is calculated, where N_{u} is the control horizon and $\triangle u(k+p|k) = u(k+p|k) - u(k+p-1|k)$. It is assumed that $\triangle u(k+p|k) = 0$ for $p \geq N_{\mathrm{u}}$. The objective of the algorithm is to minimise differences between the reference trajectory $y^{\mathrm{ref}}(k+p|k)$ and predicted values of the output $\hat{y}(k+p|k)$ over the prediction horizon N, $N \geq N_{\mathrm{u}}$. Constraints are usually imposed on input and output variables. Future control increments (1) are determined from the following MPC optimisation task (hard output constraints are used for simplicity of presentation)

$$\begin{aligned}
&\min_{\triangle \boldsymbol{u}(k)} \left\{ \sum_{p=1}^{N} (y^{\mathrm{ref}}(k+p|k) - \hat{y}(k+p|k))^2 + \lambda \sum_{p=0}^{N_{\mathrm{u}}-1} (\triangle u(k+p|k))^2 \right\} \\
&\text{subject to} \qquad (2) \\
&u^{\min} \leq u(k+p|k) \leq u^{\max},\ p = 0, \ldots, N_{\mathrm{u}} - 1 \\
&-\triangle u^{\max} \leq \triangle u(k+p|k) \leq \triangle u^{\max},\ p = 0, \ldots, N_{\mathrm{u}} - 1 \\
&y^{\min} \leq \hat{y}(k+p|k) \leq y^{\max},\ p = 1, \ldots, N
\end{aligned}$$

Only the first element of the determined sequence (1) is applied to the process, i.e. $u(k) = \triangle u(k|k) + u(k-1)$. At the next sampling instant, $k+1$, the output measurement is updated, the prediction is shifted one step forward and the whole procedure is repeated.

3 LS-SVM Hammerstein Models

Predicted values of the output variable, $\hat{y}(k+p|k)$, over the prediction horizon (i.e. for $p = 1, \ldots, N$) are calculated using the LS-SVM Hammerstein model depicted in Fig. 1. It consists of a nonlinear steady-state part in series with a linear dynamic part, $v(k)$ denotes an auxiliary signal. The steady-state part is described by the equation

$$v(k) = g(u(k))$$

where the function $g\colon \mathbb{R} \to \mathbb{R}$ is realised by the LS-SVM approximator. It has one input, n_{sv} support vectors and one output. Assuming the exponential kernel function, its output is

$$v(k) = \beta + \sum_{i=1}^{n_{\mathrm{sv}}} \alpha_i \exp\left(-\frac{(u(k) - x_{\mathrm{sv},i})^2}{\sigma^2}\right) \tag{3}$$

where the quantities $x_{\mathrm{sv},i}$ define support vectors (since the model has one input, the quantities $x_{\mathrm{sv},i}$ are scalars), α_i and β are model parameters determined during training. The linear part of the Hammerstein model is described by the difference equation

$$\boldsymbol{A}(q^{-1})y(k) = \boldsymbol{B}(q^{-1})v(k)$$

where polynomials are

$$\begin{aligned}\boldsymbol{A}(q^{-1}) &= 1 + a_1 q^{-1} + \ldots + a_{n_{\mathrm{A}}} q^{-n_{\mathrm{A}}} \\ \boldsymbol{B}(q^{-1}) &= b_\tau q^{-\tau} + \ldots + b_{n_{\mathrm{B}}} q^{-n_{\mathrm{B}}}\end{aligned}$$

The backward shift operator is denoted by q^{-1}, integers n_{A}, n_{B}, τ define the order of dynamics, $\tau \leq n_{\mathrm{B}}$. The output of the linear part of the model is

$$y(k) = \sum_{l=\tau}^{n_{\mathrm{B}}} b_l v(k-l) - \sum_{l=1}^{n_{\mathrm{A}}} a_l y(k-l) \tag{4}$$

The output of the LS-SVM Hammerstein model can be expressed as a function of input and output signal values at previous sampling instants

$$y(k) = f(\boldsymbol{x}(k)) = f(u(k-\tau), \ldots, u(k-n_{\mathrm{B}}), y(k-1), \ldots, y(k-n_{\mathrm{A}})) \tag{5}$$

From Eq. (3) and Eq. (4) one has

$$y(k) = \sum_{l=\tau}^{n_{\mathrm{B}}} b_l \left(\beta + \sum_{i=1}^{n_{\mathrm{sv}}} \alpha_i \exp\left(-\frac{(u(k-l) - x_{\mathrm{sv},i})^2}{\sigma^2}\right)\right) - \sum_{l=1}^{n_{\mathrm{A}}} a_l y(k-l) \tag{6}$$

Identification methods for LS-SVM Hammerstein models are discussed in [3]. An example application of the LS-SVM Hammerstein structure for modelling of a proton exchange membrane fuel cell stack is given in [6].

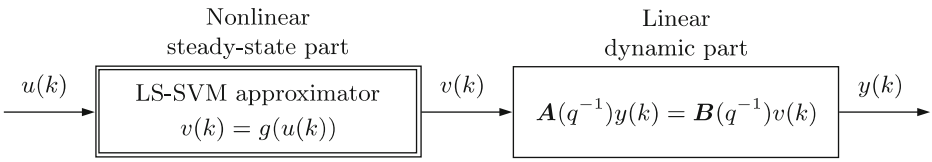


Fig. 1. The structure of the LS-SVM Hammerstein model

4 MPC-NPL Algorithm with Simple Linearisation for the Current Operating Point

The LS-SVM Hammerstein model can be directly used for prediction. Using Eq. (6) recurrently, one obtains consecutive predictions for $p = 1, \ldots, N$

$$\begin{aligned}\hat{y}(k+p|k) = &\sum_{l=1}^{I_{\mathrm{uf}}(p)} b_l \left(\beta + \sum_{i=1}^{n_{\mathrm{sv}}} \alpha_i \exp\left(-\frac{(u(k-l+p|k) - x_{\mathrm{sv},i})^2}{\sigma^2}\right)\right) \\ &\sum_{l=I_{\mathrm{uf}}(p)+1}^{n_{\mathrm{B}}} b_l \left(\beta + \sum_{i=1}^{n_{\mathrm{sv}}} \alpha_i \exp\left(-\frac{(u(k-l+p) - x_{\mathrm{sv},i})^2}{\sigma^2}\right)\right) \\ &- \sum_{l=1}^{I_{\mathrm{yf}}(p)} a_l \hat{y}(k-l+p|k) - \sum_{l=I_{\mathrm{yf}}(p)+1}^{n_{\mathrm{A}}} a_l y(k-l+p) + d(k) \end{aligned} \tag{7}$$

where $I_{\mathrm{uf}}(p) = \max(\min(p-\tau+1, n_{\mathrm{B}}-\tau+1), 0)$, $I_{\mathrm{yf}}(p) = \min(p-1, n_{\mathrm{A}})$, $d(k)$ is the estimation of the unmeasured disturbance [17]. Because the predictions are nonlinear functions of the decision variables of the MPC algorithm (1), the optimisation problem (2) is nonlinear. A straightforward idea exploited in the MPC algorithm with Nonlinear Prediction and Linearisation (MPC-NPL), the general idea of which is described in [10,17], is to use for prediction a linear approximation of the model for the current operating point. Using the Taylor series expansion formula and Eq. (5), the linearised model is

$$y(k) = f(\bar{\boldsymbol{x}}(k)) + \sum_{l=\tau}^{n_{\mathrm{B}}} b_l(k)(u(k-l) - \bar{u}(k-l)) - \sum_{l=1}^{n_{\mathrm{A}}} a_l(k)(y(k-l) - \bar{y}(k-l))$$

where the vector $\bar{\boldsymbol{x}}(k) = [\bar{u}(k-\tau)\ldots\bar{u}(k-n_{\mathrm{B}})\ \bar{y}(k-1)\ldots\bar{y}(k-n_{\mathrm{A}})]^{\mathrm{T}}$ consists of measurements and defines the current operating point. Using Eq. (6), one has

$$\begin{aligned} a_l(k) &= -\left.\frac{\partial f(\boldsymbol{x}(k))}{\partial y(k-l)}\right|_{\boldsymbol{x}(k)=\bar{\boldsymbol{x}}(k)} = a_l \\ b_l(k) &= \left.\frac{\partial f(\boldsymbol{x}(k))}{\partial u(k-l)}\right|_{\boldsymbol{x}(k)=\bar{\boldsymbol{x}}(k)} \\ &= b_l \sum_{i=1}^{n_{\mathrm{sv}}} \alpha_i \exp\left(-\frac{(\bar{u}(k-l) - x_{\mathrm{sv},i})^2}{\sigma^2}\right) \frac{2(x_{\mathrm{sv},i} - \bar{u}(k-l))}{\sigma^2} \end{aligned}$$

As derived in [9], when the linearised model is used for prediction, output predictions $\hat{\boldsymbol{y}}(k) = [\hat{y}(k+1|k) \ldots \hat{y}(k+N|k)]^{\mathrm{T}}$ are linear functions of future control increments

$$\hat{\boldsymbol{y}}(k) = \boldsymbol{G}(k)\triangle\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \tag{8}$$

The matrix $\boldsymbol{G}(k)$ of dimensionality $N \times N_{\mathrm{u}}$ contains step-response coefficients of the linearised model, the free trajectory $\boldsymbol{y}^0(k) = \left[y^0(k+1|k) \ldots y^0(k+N|k)\right]^{\mathrm{T}}$ depends only on the past. This trajectory is calculated from Eq. (7) assuming that $u(k+p|k) = u(k-1)$ for $p \geq 0$ and $\hat{y}(k+p|k) = y^0(k+p|k)$ for $p \geq 1$.

Thanks to using the prediction equation (8), the MPC optimisation problem (2) becomes the quadratic programming task

$$\min_{\triangle\boldsymbol{u}(k)} \left\{ J(k) = \left\| \boldsymbol{y}^{\mathrm{ref}}(k) - \boldsymbol{G}(k)\triangle\boldsymbol{u}(k) - \boldsymbol{y}^0(k) \right\|^2 + \|\triangle\boldsymbol{u}(k)\|^2_{\boldsymbol{\Lambda}} \right\}$$

subject to

$$\boldsymbol{u}^{\min} \leq \boldsymbol{J}\triangle\boldsymbol{u}(k) + \boldsymbol{u}(k-1) \leq \boldsymbol{u}^{\max}$$
$$-\triangle\boldsymbol{u}^{\max} \leq \triangle\boldsymbol{u}(k) \leq \triangle\boldsymbol{u}^{\max}$$
$$\boldsymbol{y}^{\min} \leq \boldsymbol{G}(k)\triangle\boldsymbol{u}(k) + \boldsymbol{y}^0(k) \leq \boldsymbol{y}^{\max}$$

In the quadratic programming problem $\boldsymbol{y}^{\mathrm{ref}} = \left[y^{\mathrm{ref}}(k+1|k) \ldots y^{\mathrm{ref}}(k+N|k)\right]^{\mathrm{T}}$, $\boldsymbol{y}^{\min} = \left[y^{\min} \ldots y^{\min}\right]^{\mathrm{T}}$ and $\boldsymbol{y}^{\max} = \left[y^{\max} \ldots y^{\max}\right]^{\mathrm{T}}$ are vectors of length N, $\boldsymbol{u}^{\min} = \left[u^{\min} \ldots u^{\min}\right]^{\mathrm{T}}$, $\boldsymbol{u}^{\max} = \left[u^{\max} \ldots u^{\max}\right]^{\mathrm{T}}$, $\triangle\boldsymbol{u}^{\max} = \left[\triangle u^{\max} \ldots \triangle u^{\max}\right]^{\mathrm{T}}$ and $\boldsymbol{u}(k-1) = \left[u(k-1) \ldots u(k-1)\right]^{\mathrm{T}}$ are vectors of length N_{u}, the diagonal matrix $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda, \ldots, \lambda)$ and the lower triangular matrix $\boldsymbol{J}$ are of dimensionality $N_{\mathrm{u}} \times N_{\mathrm{u}}$.

5 MPC-NPLTP Algorithm with Linearisation along the Trajectory

In the MPC-NPL algorithm the model is linearised once at each sampling instant k for the current operating point. The same linear approximation is used for prediction over the whole prediction horizon. In the MPC algorithm with Nonlinear Prediction and Linearisation along the Predicted Trajectory (MPC-NPLPT), the rudimentary description of which is given in [8], not the model but the predicted output trajectory is linearised. Linearisation can be carried out once for an assumed trajectory or in an iterative manner a few times at each sampling instant. In the n^{th} internal iteration linearisation along the input trajectory $\boldsymbol{u}^{n-1}(k) = \left[u^{n-1}(k|k) \ldots u^{n-1}(k+N_{\mathrm{u}}-1|k)\right]^{\mathrm{T}}$ found in the previous internal iteration is calculated. Using the Taylor series expansion formula, one obtains a linear approximation of the predicted nonlinear output trajectory $\hat{\boldsymbol{y}}^n(k) = [\hat{y}^n(k+1|k) \ldots \hat{y}^n(k+N|k)]^{\mathrm{T}}$ calculated at the current internal iteration

$$\hat{\boldsymbol{y}}^n(k) = \hat{\boldsymbol{y}}^{n-1}(k) + \boldsymbol{H}^n(k)(\boldsymbol{u}^n(k) - \boldsymbol{u}^{n-1}(k)) \tag{9}$$

where the vectors $\boldsymbol{u}^n(k)$ and $\hat{\boldsymbol{y}}^{n-1}(k)$ are similar to the vector $\boldsymbol{u}^{n-1}(k)$ and $\hat{\boldsymbol{y}}^n(k)$, respectively. The nonlinear trajectory $\hat{\boldsymbol{y}}^{n-1}(k)$ is calculated for the input trajectory $\boldsymbol{u}^{n-1}(k)$ recurrently from Eq. (7) assuming $u(k+p|k) = u^{n-1}(k+p|k)$ and $\hat{y}(k+p|k) = \hat{y}^{n-1}(k+p|k)$. The matrix $\boldsymbol{H}^n(k)$ is of dimensionality $N \times N_\mathrm{u}$ and consists of partial derivatives of the predicted output trajectory $\hat{\boldsymbol{y}}^{n-1}(k)$ with respect to the input trajectory $\boldsymbol{u}^{n-1}(k)$

$$\boldsymbol{H}^n(k) = \frac{\mathrm{d}\hat{\boldsymbol{y}}^{n-1}(k)}{\mathrm{d}\boldsymbol{u}^{n-1}(k)} = \begin{bmatrix} \dfrac{\partial \hat{y}^{n-1}(k+1|k)}{\partial u^{n-1}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{n-1}(k+1|k)}{\partial u^{n-1}(k+N_\mathrm{u}-1|k)} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial \hat{y}^{n-1}(k+N|k)}{\partial u^{n-1}(k|k)} & \cdots & \dfrac{\partial \hat{y}^{n-1}(k+N|k)}{\partial u^{n-1}(k+N_\mathrm{u}-1|k)} \end{bmatrix}$$

Taking into account the structure of the model, form Eq. (7) one has

$$\begin{aligned} \frac{\partial \hat{y}^{n-1}(k+p|k)}{\partial u^{n-1}(k+r|k)} = & \sum_{l=1}^{I_\mathrm{uf}(p)} b_l \sum_{i=1}^{n_\mathrm{sv}} \alpha_i \exp\left(-\frac{(u^{n-1}(k-l+p|k) - x_{\mathrm{sv},i})^2}{\sigma^2}\right) \\ & \times \frac{2(x_{\mathrm{sv},i} - u^{n-1}(k-l+p|k))}{\sigma^2} \frac{\partial u^{n-1}(k-l+p|k)}{\partial u^{n-1}(k+r|k)} \\ & - \sum_{l=1}^{I_\mathrm{uf}(p)} a_l \frac{\partial \hat{y}^{n-1}(k-l+p|k)}{\partial u^{n-1}(k+r|k)} \end{aligned}$$

Using Eq. (9), the optimisation problem (2) becomes the following quadratic programming problem

$$\begin{aligned} \min_{\triangle \boldsymbol{u}^n(k)} \Big\{ & \big\| \boldsymbol{y}^\mathrm{ref}(k) - \hat{\boldsymbol{y}}^{n-1}(k) - \boldsymbol{H}^n(k)\boldsymbol{J}\triangle\boldsymbol{u}^n(k) \\ & - \boldsymbol{H}^n(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{n-1}(k)) \big\|^2 + \|\triangle\boldsymbol{u}^n(k)\|^2_{\boldsymbol{\Lambda}} \Big\} \end{aligned}$$

subject to

$$\begin{aligned} & \boldsymbol{u}^\mathrm{min} \le \boldsymbol{J}\triangle\boldsymbol{u}^n(k) + \boldsymbol{u}(k-1) \\ & -\triangle\boldsymbol{u}^\mathrm{max} \le \triangle\boldsymbol{u}^n(k) \le \triangle\boldsymbol{u}^\mathrm{max} \\ & \boldsymbol{y}^\mathrm{min} \le \hat{\boldsymbol{y}}^{n-1}(k) + \boldsymbol{H}^n(k)\boldsymbol{J}\triangle\boldsymbol{u}^n(k) + \boldsymbol{H}^n(k)(\boldsymbol{u}(k-1) - \boldsymbol{u}^{n-1}(k)) \le \boldsymbol{y}^\mathrm{max} \end{aligned}$$

If the the operating point does not change significantly, it would be sufficient to carry out only one internal iteration. Internal iterations are continued if

$$\sum_{p=0}^{N_0} (y^\mathrm{ref}(k-p) - y(k-p))^2 \ge \delta_\mathrm{y}$$

If $\big\|\triangle\boldsymbol{u}^n(k) - \triangle\boldsymbol{u}^{n-1}(k)\big\|^2 < \delta_\mathrm{u}$ or $n > n_\mathrm{max}$ internal iterations are terminated. The quantities δ_u, δ_y, N_0 and n_max are adjusted by the user.

In the MPC algorithm with Nonlinear Prediction and Linearisation along the Trajectory (MPC-NPLT), the general idea of which is given in [7], a linear

approximation of the nonlinear output trajectory $\hat{\boldsymbol{y}}(k)$ along an input trajectory $\boldsymbol{u}^{\mathrm{traj}}(k)$ is calculated only once. In such a case the linearisation formula (9) becomes

$$\hat{\boldsymbol{y}}(k) = \hat{\boldsymbol{y}}^{\mathrm{traj}}(k) + \boldsymbol{H}(k)(\boldsymbol{u}(k) - \boldsymbol{u}^{\mathrm{traj}}(k))$$

For linearisation the input trajectory $\boldsymbol{u}^{\mathrm{traj}}(k) = [u(k-1) \ldots u(k-1)]^{\mathrm{T}}$ can be used, which leads to the MPC-NPLT$_{\boldsymbol{u}(k-1)}$ algorithm, or the trajectory $\boldsymbol{u}^{\mathrm{traj}}(k) = [u(k|k-1) \ldots u(k+N_{\mathrm{u}}-3|k-1)\, u(k+N_{\mathrm{u}}-2|k-1)\, u(k+N_{\mathrm{u}}-2|k-1)]^{\mathrm{T}}$, which leads to the MPC-NPLT$_{\boldsymbol{u}(k|k-1)}$ algorithm.

6 Simulation Results

The linear dynamic part of the Hammerstein system [4] is defined by polynomials

$$\boldsymbol{A}(q^{-1}) = 1 - 1.2q^{-1} + 0.9q^{-2},\ \boldsymbol{B}(q^{-1}) = 1.7q^{-1} - q^{-2} \tag{10}$$

Its nonlinear steady-state part $v(k) = 2\mathrm{sign}(u(k))\sqrt{|u(k)|}$ is shown in Fig. 2.

The following MPC algorithms are compared:

a) the classical linear MPC algorithm based on the linear model,
b) the nonlinear MPC-NPL algorithm,
c) the nonlinear MPC-NPLT$_{\boldsymbol{u}(k-1)}$ and MPC-NPLT$_{\boldsymbol{u}(k|k-1)}$ algorithms,
d) the nonlinear MPC-NPLPT algorithm,
e) the "ideal" MPC-NO algorithm with on-line nonlinear optimisation.

The linear MPC algorithm uses the model (10). All nonlinear MPC algorithms use the Hammerstein model with the linear dynamic part (10), the steady-state nonlinear part is the LS-SVM approximator with 100 support vectors (Fig. 2). Parameters of all MPC algorithms are: $N = 10$, $N_{\mathrm{u}} = 3$, $\lambda = 5$, $u^{\min} = -1.5$, $u^{\max} = 1.5$. Additional parameters of the MPC-NPLPT algorithm are: $\delta_{\mathrm{u}} = \delta_{\mathrm{y}} = 10^{-1}$, $N_0 = 2$, $n_{\max} = 5$.

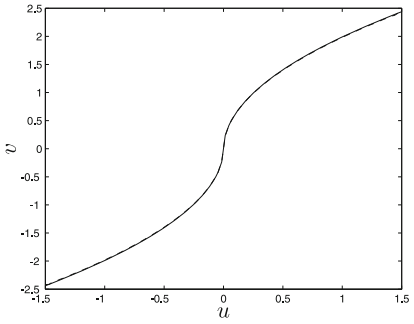


Fig. 2. The characteristics $v(k) = g(u(k))$ of the steady-state part of the process (*solid line*) and its LS-SVM approximation (*dashed line*)

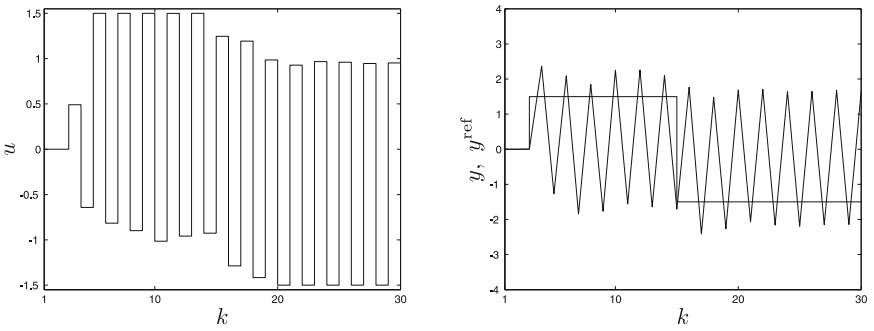


Fig. 3. Simulation results of the classical linear MPC algorithm

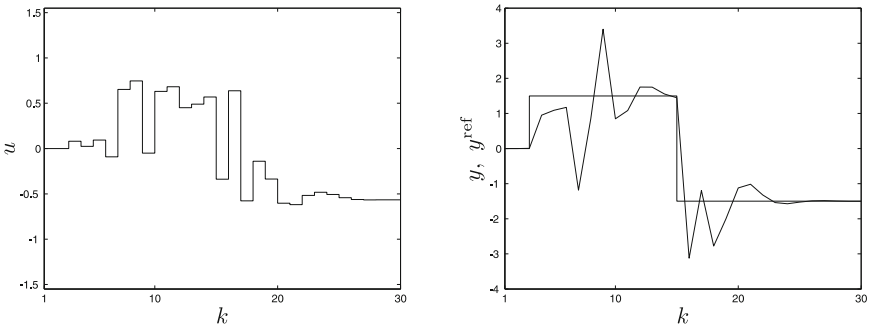


Fig. 4. Simulation results of the MPC-NPL algorithm

Because the process is significantly nonlinear, the linear MPC algorithm does not work properly as depicted in Fig. 3. It also turns out that the simple linearisation for the current operating point used in the MPC-NPL algorithm is inefficient as shown in Fig. 4. The MPC-NPLT$_{\boldsymbol{u}(k-1)}$ and MPC-NPLT$_{\boldsymbol{u}(k|k-1)}$ algorithms with linearisation along an assumed future input trajectory are much better as shown in Fig. 5, but still the control accuracy is not satisfying. Finally, Fig. 6 compares trajectories obtained in the MPC-NPLPT algorithm and in the "ideal" but computationally demanding MPC-NO approach with nonlinear optimisation repeated at each sampling instant. For the considered process the MPC-NPLPT algorithm gives trajectories very close to those obtained in the MPC-NO approach. Because the process is significantly nonlinear, the iterative linearisation along the predicted trajectory is necessary rather than single model or trajectory linearisation used in MPC-NPL and MPC-NPLT algorithms.

Table 1 shows accuracy (in terms of Sum of Squared Errors, SSE) and computational load (in terms of floating point operations, MFLOPS) of compared nonlinear algorithms. The MPC-NPLPT algorithm is many times less computationally demanding than the MPC-NO approach.

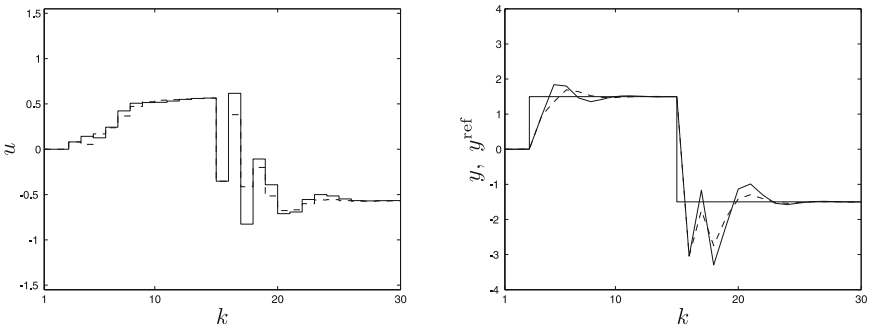


Fig. 5. Simulation results: the MPC-NPLT$_{\boldsymbol{u}(k-1)}$ algorithm (*solid line*) and the MPC-NPLT$_{\boldsymbol{u}(k|k-1)}$ algorithm (*dashed line*)

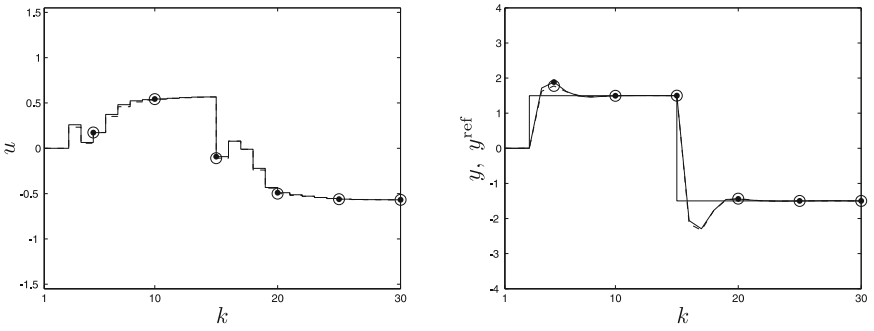


Fig. 6. Simulation results: the MPC-NPLPT algorithm (*dashed line with circles*) vs. the MPC-NO algorithm (*solid line with dots*)

Table 1. Accuracy (SSE) and computational load (MFLOPS) of compared nonlinear MPC algorithms based on the same LS-SVM Hammerstein model

Algorithm	Optimisation	SSE	MFLOPS
MPC-NPL	Quadratic	28.48	0.49
MPC-NPLT$_{\boldsymbol{u}(k-1)}$	Quadratic	18.40	1.18
MPC-NPLT$_{\boldsymbol{u}(k|k-1)}$	Quadratic	15.87	1.18
MPC-NPLPT	Quadratic	12.53	1.68
MPC-NO	Nonlinear	12.45	16.10

7 Conclusions

The described nonlinear MPC algorithms for LS-SVM Hammerstein models are computationally efficient because they require solving on-line a quadratic programming problem or a series of such problems, nonlinear optimisation is not

necessary. For the considered process the iterative linearisation along the predicted trajectory used in the MPC-NPLPT algorithm gives the best results.

Acknowledgement. The work presented in this paper was supported by Polish national budget funds for science.

References

1. Eskinat, E., Johnson, S., Luyben, W.L.: Use of Hammerstein models in identification of nonlinear systems. AIChE Journal 37, 255–268 (1991)
2. Fruzzetti, K.P., Palazoğlu, A., McDonald, K.A.: Nonlinear model predictive control using Hammerstein models. Journal of Process Control 7, 31–41 (1997)
3. Goethals, I., Pelckmans, K., Suykens, J.A.K., De Moor, B.: Identification of MIMO Hammerstein models using Least Squares Support Vector Machines. Automatica 41, 1263–1272 (2005)
4. Hong, X., Iplikci, S., Chen, S., Warwick, K.: A model-based PID controller for Hammerstein systems using B-spline neural networks. International Journal of Adaptive Control and Signal Processing, doi: 10.1002/acs.2293
5. Janczak, A.: Identification of nonlinear systems using neural networks and polynomial models: block oriented approach. Springer, London (2004)
6. Li, C.H., Zhu, X.J., Cao, G.Y., Sui, S., Hu, M.R.: Identification of the Hammerstein model of a PEMFC stack based on Least Squares Support Vector Machines. Journal of Power Sources 175, 303–316 (2008)
7. Ławryńczuk, M.: On-Line Trajectory-Based Linearisation of Neural Models for a Computationally Efficient Predictive Control Algorithm. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2012, Part I. LNCS, vol. 7267, pp. 126–134. Springer, Heidelberg (2012)
8. Ławryńczuk, M.: On improving accuracy of computationally efficient nonlinear predictive control based on neural models. Chemical Engineering Science 66, 5253–5267 (2011)
9. Ławryńczuk, M.: Suboptimal nonlinear predictive control based on multivariable neural Hammerstein models. Applied Intelligence 32, 173–192 (2010)
10. Ławryńczuk, M.: A family of model predictive control algorithms with artificial neural networks. International Journal of Applied Mathematics and Computer Science 17, 217–232 (2007)
11. Maciejowski, J.M.: Predictive control with constraints. Prentice Hall, Harlow (2002)
12. Schölkopf, B., Smola, A.: Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT, Cambridge (2001)
13. Smith, J.G., Kamat, S., Madhavan, K.P.: Modeling of pH process using wavenet based Hammerstein model. Journal of Process Control 17, 551–561 (2007)
14. Suykens, J.A.K., Van Gestel, T., De Brabanter, J., De Moor, B., Vandewalle, J.: Least squares support vector machines. World Scientific, Singapore (2002)
15. Qin, S.J., Badgwell, T.A.: A survey of industrial model predictive control technology. Control Engineering Practice 11, 733–764 (2003)
16. Tatjewski, P.: Supervisory predictive control and on-line set-point optimization. International Journal of Applied Mathematics and Computer Science 20, 483–495 (2010)
17. Tatjewski, P.: Advanced control of industrial processes, Structures and algorithms. Springer, London (2007)

Particle Swarm Optimization with Transition Probability for Timetabling Problems

Hitoshi Kanoh[1] and Satoshi Chen[2]

[1] University of Tsukuba, Faculty of Engineering, Information and Systems, Tsukuba, Japan
kanoh@cs.tsukuba.ac.jp
[2] University of Tsukuba, Graduate School of Systems and Information Engineering, Japan

Abstract. In this paper, we propose a new algorithm to solve university course timetabling problems using a Particle Swarm Optimization (PSO). PSOs are being increasingly applied to obtain near-optimal solutions to many numerical optimization problems. However, it is also being increasingly realized that PSOs do not solve constraint satisfaction problems as well as other meta-heuristics do. In this paper, we introduce transition probability into PSO to settle this problem. Experiments using timetables of the University of Tsukuba showed that this approach is a more effective solution than an Evolution Strategy.

Keywords: Particle swarm optimization, university course timetable, constraint satisfaction problem, transition probability.

1 Introduction

University course timetabling problems (UCTPs) are search problems involving assigning timeslots to subjects so that existing constraints are satisfied [1-3]. The problems are generally characterized as constraint satisfaction problems to minimize the total penalty for constraint violations and generate feasible but not optimal solutions. The problems that belong to the NP-hard class are quite complex and very difficult to solve using conventional optimization techniques. These include Simulated Annealing [4], Case-Based Reasoning [5], Graph Based Hyperheuristics [6], Ant Colony Optimization [7], Genetic Algorithms [8-10], and Particle Swarm Optimization (PSO) [11, 12].

PSO is a population-based stochastic search algorithm that is inspired by the social interaction behavior of birds flocking and fish schooling [13, 14]. The first version of PSO was intended to handle only continuous optimization problems [15-17]. As many optimization problems are defined in the discrete space, research on extending the PSO to solve discrete combinatorial optimization problems (COPs) and constraint satisfaction problems (CSPs) has become an attractive subject in recent years.

Although various PSO algorithms have been proposed, they generally do not perform satisfactorily [18]. Compared with the other meta-heuristics for COPs and CSPs, their performance is not competitive. On the other hand, the hybrid PSO algorithms outperform the pure PSO approaches. However, as these algorithms are generally designed for specific problems, their structures are more complicated, and they are hard to apply to other COPs [18].

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 256–265, 2013.

In this paper, we propose a new PSO algorithm to solve UCTPs. Although pure and hybrid PSOs for timetabling problems have been proposed [11, 12, 19, 20], few general-use PSOs for UCTPs in the real world have been found. In the proposed method, transition probability is introduced into pure PSO, and a phenotype (timetable) can be used directly as a representation of a solution. It is easy to apply this technique to other COPs and CSPs.

In the following sections, we first discuss the objective problem and related works. Next, we describe the algorithm of the proposed method in detail. Finally, we give the results of experiments using real timetables in Japan and show that this approach is a more effective solution than an Evolution Strategy (ES).

2 Problem Description

2.1 Objective Problem

The timetables to be treated in this paper are constructed for the undergraduate courses of the College of Information Science in the University of Tsukuba, Japan. The college includes three specialized areas, four grades, and three terms. Though there are five interconnected timetables, we need to construct the timetables at the same time. Figure 1 shows an example of the objective timetable.

The problem is defined using four terms: $Problem = (S, T, C, W)$, where $S = \{S_1, \ldots, S_n\}$ is a set of subjects; $T = \{T_1, \ldots, T_m\}$ is a set of timeslots for allocation and $T_j =$ (term, day of the week, time period); $C = \{C_1, \ldots, C_q\}$ is a set of constraints; and $W = \{W_1, \ldots, W_q\}$ is a set of weights (i.e., penalties) for the constraints. In addition, each subject has attributes such as a subject name, an area, a grade, the number of time periods, whether it is compulsory or an elective, the teacher in charge, and whether computers are used.

Each timeslot consists of a term, a day of the week, and a time period, where the term is 1 to 3, the day of the week is Monday to Friday, and the time period is 1 to 6. There are 450 ($=m$) timeslots, 112 ($=n$) subjects, and 72 teachers in the problem treated here, but most subjects have lectures for two time periods (timeslots) in a week. These are henceforth called two-period subjects.

	Mon	Tue	Wed	Thu	Fri
1		Algebra II	Statictics	Signal proc.	Physics
2	Speech recog.				
3			Experi-ment	OS II	
4		Image recog.		Database	Experi-ment
5					
6					

× 3 terms × 5 timetables

Fig. 1. Example of timetable. The objective problem has 450 timeslots, 112 subjects, and 72 teachers.

2.2 Constraints in Search

The constraints we treat are classified as hard or soft. Hard constraints are those to which a timetable has to adhere in order to be feasible. Soft constraints are those that should preferably be satisfied, but which can be acceptably broken with a penalty associated with their violation. The 12 constraints shown in Table 1 are applied to the problems. In this table, C_1 to C_5 are hard constraints and C_6 to C_{12} are soft constraints with weights W_6 to W_{12}.

Table 1. Constraints and their weights applied (C_1 to C_5 are hard and the others are soft constraints). Weight of C_8 depends on the number of subjects in violation.

Symbol	Constraint	Weight
C_1	Each teacher can take only one class at the same time.	NA
C_2	Subjects that are assigned in their order of priority must keep their orders.	NA
C_3	More than one subject must not be allocated to the same time slot in the same timetable.	NA
C_4	Only one subject can be allocated to a computer room during the same time period.	NA
C_5	The same subjects (two-period subjects) must be allocated to sequential time periods on a day.	NA
C_6	Compulsory subjects of one grade and the other grades should not be allocated to the same time slot.	10
C_7	The subjects taught over two terms should be allocated to the same time period and day of the week.	10
C_8	The number of subjects per day should be equalized. Subjects should not concentrate on a specific day of the week.	5 to 15
C_9	More than one subject should not be allocated to the same time slot in the timetables for specialized areas.	3
C_{10}	Compulsory subjects should not be allocated to more than four time periods per day.	2
C_{11}	Lunch break should not be between subjects with lectures over two or more time periods in a row.	2
C_{12}	Subjects should not be allocated to the sixth time period.	1

2.3 Particle Swarm Optimization

Particle swarm optimization (PSO) is a stochastic search algorithm for problem solving that is inspired by simulations of the swarm behavior of birds flocking [13, 14]. In PSO, a number of individuals, called particles, are placed in the search space of a problem, and each particle evaluates its fitness, i.e. the objective function, at its current location. The coordinates of each particle represent a possible solution associated

with two vectors: the position and the velocity vectors. Each particle adjusts its flying in accordance with its own and its companions' flying experience.

The each particle is a point in N-dimensional search space. The position and velocity of i-th particle are represented as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iN})$ and $v_i = (v_{i1}, v_{i2}, \ldots, v_{iN})$. The best previous position, i.e. giving the best fitness value, of particle i is represented as $pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iN})$ and the best previous position among all particles in the population is represented as $gbest = (gbest_1, gbest_2, \ldots, gbest_N)$. Each particle updates its position in accordance with formula (1):

$$\begin{aligned} v_i(k+1) &= wv_i(k) + c_1r_1(pbest_i(k) - x_i(k)) + c_2r_2(gbest(k) - x_i(k)) \\ x_i(k+1) &= x_i(k) + v_i(k+1) \end{aligned} \tag{1}$$

Where, k is the iteration index, c_1 and c_2 are constants, r_1 and r_2 are uniform random numbers in the range of [0, 1], and w is the inertia weight. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

2.4 Related Works

Fen et. al. [12] proposed a hybrid PSO algorithm to solve the university course timetabling problem. The proposed approach uses PSO to find the position of rooms and timeslots using a suitable objective function and the constraint-based reasoning (CBR). This algorithm has been validated with other hybrid algorithms (hybrid PSO with local search and hybrid genetic algorithm with CBR) using real world data. However, each particle is mainly updated by CBR, and small problems fit this method. This paper treats a problem with only 18 time slots and there is no soft constraint.

Tassopoulos et. al. [20] proposed a new adaptive algorithm based on PSO and applied it to the high school timetabling problem. The proposed PSO algorithm is used to create feasible and very efficient timetables for high schools in Greece. However, the university timetable is more complicated in terms of constraints and the attributes of a subject than the school timetable. The particle encoding in this paper may not be useful for the university timetabling problem.

Shiau [19] proposed a novel meta-heuristic algorithm that is based on PSO for the university course scheduling problem (instructors and class scheduling arrangements). The algorithms were tested using the timetabling data from a typical university in Taiwan. This algorithm also outperforms the genetic algorithm proposed in the literature. However, each particle is updated on the basis of continuous PSO formulas and local search. This method can be applied to only small problems in experiments.

3 Proposed Method

3.1 Particle Representation

Figure 2 shows the structure and an example of a particle in the proposed method. We use direct coding for the particle representation, where each coordinate axis corresponds to a subject and each component of a position vector corresponds to a timeslot

allocated to the subject. PSO assigns timeslots to subjects so as to minimize the objective function. As mentioned in Section 2.1, five timetables have to be constructed at the same time, so a particle consists of five timetables as shown in Fig. 2. The operation of PSO is separately performed on each timetable. A two-period subject uses two separate timeslots, and they are treated as independent timeslots.

Let x_{ij} be the j-th component of particle i. In Figs. 1 and 2, for example, x_{11}=(1, Fri, 1), x_{12}=(1, Tue, 1) and x_{13}=(1, Tue, 2) mean that Physics is assigned to the first time period on Friday in term 1 and Algebra II is assigned to the first and second time periods on Tuesday in term 1.

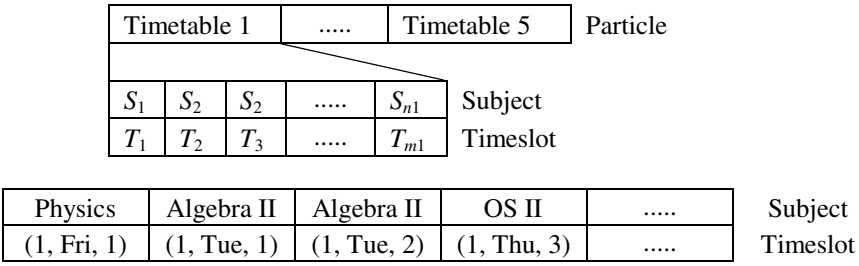


Fig. 2. Structure and example of particle (solution). T_j = (term, day of the week, time period). $n1$ and $m1$ are the numbers of subjects and timeslots allocated in Timetable 1, respectively. Since there are many two-period subjects, $n1$ is less than $m1$.

3.2 Objective Function

The objective function of each particle in the population is measured by the degree to which the timetables in the particle meet the constraints. A timeslot is assigned to a subject so as to satisfy all the hard constraints in the initial population generation and the particle flying (see Procedure update-position in section 3.4). Therefore, only soft constraints are taken into consideration as an objective function. The objective function value (fitness) of the i-th particle f_i can be expressed by formula (2), where M_{il} is the number of times that the i-th particle violates the constraint C_l with penalty W_l. In addition, the objective function is calculated through five timetables.

$$f_i = \sum_{l=1}^{q} M_{il} W_l \tag{2}$$

3.3 Transition Probability

Transition probability to a subject is the probability that a timeslot of the subject on a particle will rewrite or exchange. This probability consists of three connected probabilities: its own empty timeslot P_{own}, a timeslot of the subject on the personal best particle P_{pbest}, and a timeslot of the subject on the global best particle P_{gbest} (P_{own}+P_{pbest}+P_{gbest}=1). Each probability can be defined by formula (3); where w, c_1, and c_2 are constants defined by experiments, r_1 and r_2 are uniform random numbers in the range of [0, 1].

$$P_{own} = \frac{w}{w + c_1 r_1 + c_2 r_2}$$
$$P_{pbest} = \frac{c_1 r_1}{w + c_1 r_1 + c_2 r_2} \tag{3}$$
$$P_{gbest} = \frac{c_2 r_2}{w + c_1 r_1 + c_2 r_2}$$

In addition, the calculation of the transition probability in the proposed method corresponds to the calculation of velocity in continuous PSOs.

3.4 Pseudo Code of Algorithms

The main procedure and the procedure for one update of position in the flying process of i-th particle x_i are as follows, where $T = \{T_1, \ldots, T_m\}$ is a set of timeslots (see 2.1).

```
Procedure main()
   Generate initial population randomly;
   Evaluate fitness of all particles;
   Save gbest and all pbests;
   While (terminal condition not met) {
      for (i = 1 to number of particles) {
         update-position(x_i);
         Evaluate f_i;
         Update pbest_i;
      }
      Update gbest;
   }

Procedure update-position(x_i)
   for ( j = 1 to N) {
      Calculate probabilities P_own, P_pbest, and P_gbest;
      Generate uniform random number rand ∈ [0, 1];
      if ( rand <= P_own ) {
         Timeslot T_p ∉ x_i is randomly selected from T so as
         to satisfy all hard constraints;
         x_ij is rewritten as T_p;
      }
      if ( P_own < rand <= P_own + P_pbest ) {
         if ( pbest_ij is included in x_i )
         x_ij is exchanged to x_i's component the value of
         which is pbest_ij;
      else
         x_ij is rewritten as pbest_ij;
         if ( x_ij violates one or more hard constraints )
            Last allocation is canceled;
      }
      if ( P_own + P_pbest < rand ) {
```

```
        if ( gbest_j is included in x_i )
        x_ij is exchanged to x_i's component the value of
        which is gbest_j;
     else
        x_ij is rewritten as gbest_ij;
        if ( x_ij violates one or more hard constraints )
           Last allocation is canceled;
     }
  }
```

4 Experiments

4.1 Parameter Dependence

We first examined the dependence of the total penalty on the parameters (w, c_1, and c_2) through three experiments, where the population size was 200 and the number of iterations was 500. Figure 3 shows the experimental results. Each point in the figure is the average of 10 trials using different random number sequences. The optimal values of the parameters obtained by the experiments were w=0.05, c_1=5.0, and c_2=2.0.

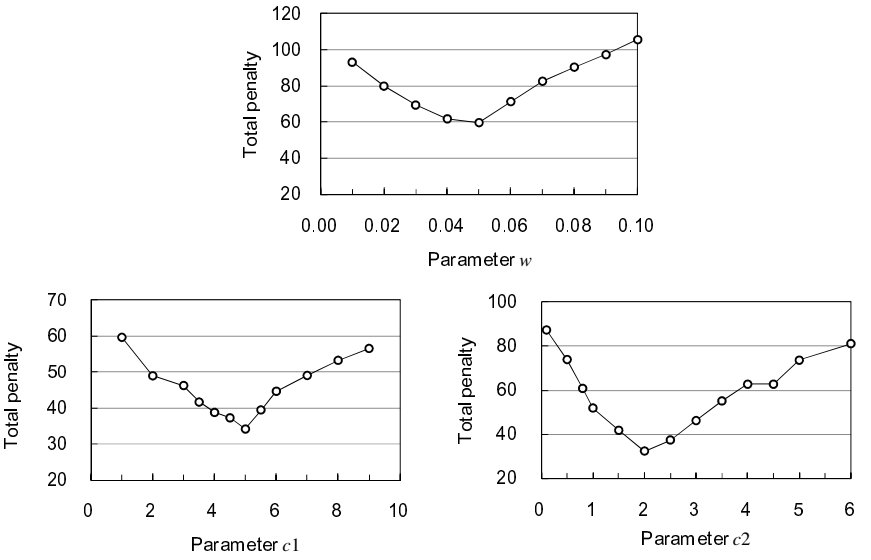


Fig. 3. Parameter dependence of the proposed method. The values of the other parameters were c_1=c_2=2.0 in the upper figure, w=0.05, c_2=2.0 in the lower left figure, and w=0.05, c_1=5.0 in the lower right figure.

4.2 Comparative Study

To evaluate the performance of the proposed method, we compared it with a (μ+λ) evolution strategy (ES). The mutation operation of the ES, “exchange”, was

performed for every timetable individually. First, we randomly selected a timeslot to which a subject is allocated. Second, we randomly selected another timeslot that includes an empty timeslot so as to satisfy all the hard constraints. Finally, we exchanged the two subjects (if the second subject is empty, this operation is "movement"). In this section, $\lambda = 2\mu$ for ES and each point in figures is the average of 10 trials.

Figure 4 shows the relationship between the population size N_I and the total penalty of the best solution in the final iteration step f_{best}. The number of iterations was 1000 for both methods. We can see that the value of f_{best} at $N_I = 200$ for PSO is better than the value of f_{best} at $N_I = 1000$ for ES.

Figure 5 shows the total penalties of the best solution in each iteration step. We set N_I of PSO = $\lambda = 1000$, so the number of fitness evaluations, i.e. computational cost, for ES was the same as that for PSO. We found that the total penalty of PSO obviously had a lower final value than ES. In addition, the total penalties of the best solution in ten trials for PSO and ES were 17 and 21, respectively.

We investigated violation of the best solutions in the final iteration of Fig. 5. Table 2 shows the number of unsatisfied constraints in the timetable, the total penalty of which is the median in ten constructed timetables for both methods. We can see the following:

- The proposed PSO did not violate high penalty soft constraints (C_6 to C_8), even if it violated the lowest penalty constraint C_{12} many times; resulting in the lower total penalty than ES.
- The ES twice violated high penalty constraints C_8, so it was difficult for one exchange or one movement to satisfy C_8 because several subjects were connected to this constraint.

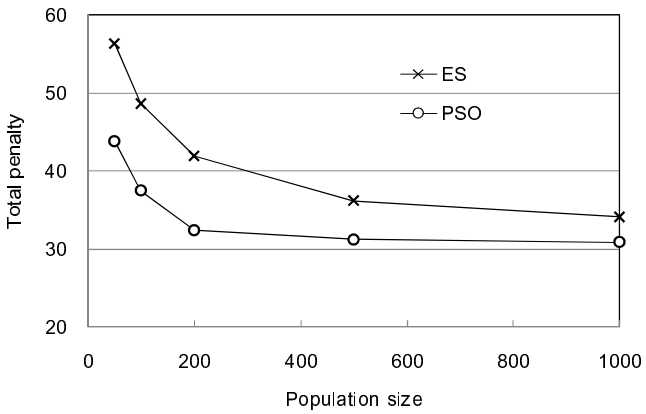


Fig. 4. Relationship between population size and total penalty. The number of iterations is 1000 for ES and PSO, and $\lambda = 2\mu$ for ES. Each point is the average of 10 trials.

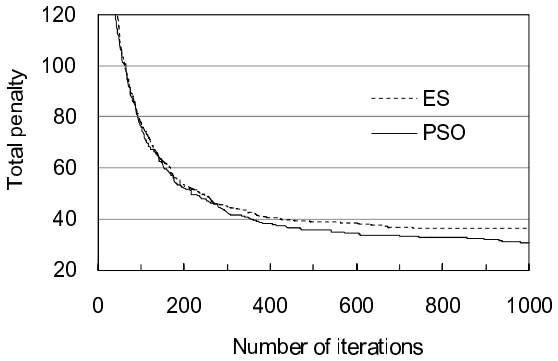


Fig. 5. Total penalty of best solution in each iteration step. Each point is the average of 10 trials. Total penalty of best solution in initial population was about 210 for both methods.

Table 2. Number of unsatisfied constraints for each method. Contents of constraints are referred to in Table 1. Number in parenthesis indicates penalty. Total penalties of PSO and ES are respectively 30 and 36 in this case.

Method	Soft constraint (penalty)						
	C_6 (10)	C_7 (10)	C_8 (*)	C_9 (3)	C_{10} (2)	C_{11} (2)	C_{12} (1)
PSO	0	0	0	0	1	5	18
ES	0	0	2	0	0	1	10

* Two penalties against C_8 in ES are 10 and 15.

5 Conclusions

In this paper, we solved university course timetabling problems using PSO with transition probability. Experiments using timetables of the University of Tsukuba showed that this approach is an effective solution. We do not, however, claim that the PSO is the best method to solve the problem. The proposed method includes a general technique that can be applied to various large-scale real-life COPs and CSPs. We estimated calculation cost using the number of fitness evaluations. To the real world problems, however, computation time is also important. In our forthcoming study, we will make a detailed comparison with genetic algorithms using both total penalty and computation time. This research was partly supported by a Grant-in Aid for Scientific Research (C) of the Japan Society for the Promotion of Science (23500169).

References

1. Schaerf, A.: A Survey of Automated Timetabling. Artificial Intelligence Review 13(2), 87–127 (1999)
2. Gröbner, M., Wilke, P., Büttcher, S.: A Standard Framework for Timetabling Problems. In: Burke, E.K., De Causmaecker, P. (eds.) PATAT 2002. LNCS, vol. 2740, pp. 24–38. Springer, Heidelberg (2003)

3. Dimopoulou, M., Miliotis, P.: An Automated University Course Timetabling System Developed in a Distributed Environment: a Case Study. European Journal of Operational Research 153, 136–147 (2004)
4. Fukushima, M.: A Hybrid Algorithm for the University Course Timetabling Problems. Journal of Japan Society for Fuzzy Theory and Intelligent Informatics 22(1), 142–147 (2010)
5. Burke, E.K., MacCarthy, B., Petrovic, S., Qu, R.: Multiple-Retrieval Case Based Reasoning for Course Timetabling Problems. Journal of the Operational Research Society 57(2), 148–162 (2006)
6. Burke, E.K., et al.: A Graph-Based Hyper-Heuristic for Educational Timetabling Problems. European Journal of Operational Research 176(1), 177–192 (2007)
7. Socha, K., Sampels, M., Manfrin, M.: Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In: Raidl, G.R., Cagnoni, S., Cardalda, J.J.R., Corne, D.W., Gottlieb, J., Guillot, A., Hart, E., Johnson, C.G., Marchiori, E., Meyer, J.-A., Middendorf, M. (eds.) EvoIASP 2003, EvoWorkshops 2003, EvoSTIM 2003, EvoROB/EvoRobot 2003, EvoCOP 2003, EvoBIO 2003, and EvoMUSART 2003. LNCS, vol. 2611, pp. 334–345. Springer, Heidelberg (2003)
8. Burke, E.K., Landa, J.D.: The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In: Recent Advances in Memetic Algorithms and Related Search Technologies, pp. 289–312. Springer (2004)
9. Lewis, R., Paechter, B.: Finding Feasible Timetables Using Group-Based Operators. IEEE Transactions on Evolutionary Computation 11(3), 397–413 (2007)
10. Kanoh, H., Sakamoto, Y.: Knowledge-Based Genetic Algorithm for University Course Timetabling Problems. International Journal of Knowledge-Based and Intelligent Engineering Systems 12(4), 283–294 (2008)
11. Qu, R., Burke, E.K.: Hybridizations within a Graph-based Hyper-heuristic Framework for University Timetabling Problems. Journal of the Operational Research Society 60, 1273–1285 (2009)
12. Fen, H.S., Safaai, D., Hashim, M., Zaiton, S.: University Course Timetable Planning Using Hybrid, Particle Swarm Optimization. In: GEC 2009, June 12-14 (2009)
13. Poli, R., Kennedy, J., Blackwell, T.: Particle Swarm Optimization: An Overview. Swarm Intell. 1, 33–57 (2007)
14. Banks, A., Vincent, J., Anyakoha, C.: A Review of Particle Swarm Optimization Part II: Hybridisation, Combinatorial, Multicriteria and Constrained Optimization, and Indicative Applications. Nat. Comput. 7, 109–124 (2008)
15. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: IEEE Int. Conf. Neural Netw., vol. 4, pp. 1942–1948 (1995)
16. Eberhart, R., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proc. 6th Int. Symp. Micro Machine Human Science, pp. 39–43 (1995)
17. Shi, Y., Eberhart, R.: A Modified Particle Swarm Optimizer. IEEE World Congr. Comput. Intell., 69–73 (1998)
18. Chen, W.N., et al.: A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems. IEEE Transactions on Evolutionary Computation 14(2), 278–300 (2010)
19. Shiau, D.F.: A Hybrid Particle Swarm Optimization for a University Course Scheduling Problem with Flexible Preferences. Expert Systems with Applications 38, 235–248 (2011)
20. Tassopoulos, I.X., Beligiannis, G.N.: Solving Effectively the School Timetabling Problem using Particle Swarm Optimization. Expert Systems with Applications 39, 6029–6040 (2012)

A Consensus Approach for Combining Multiple Classifiers in Cost-Sensitive Bankruptcy Prediction

Ning Chen[1] and Bernardete Ribeiro[2]

[1] GECAD, Instituto Superior de Engenharia do Porto, Portugal
[2] CISUC, Department of Informatics Engineering, University of Coimbra, Portugal
ningchen74@gmail.com, bribeiro@dei.uc.pt

Abstract. Bankruptcy prediction is an extremely important topic in the field of financial decision making. There has been a raising interest in studying more accurate predictive models able to provide valuable early warning before the real business failure. Recent researches suggested using the consensus of multiple classifiers for boosting the prediction performance. Yet rarely the cost of misclassification errors is considered in the literature of consensus decision making. In this paper we investigate the performance of classifier ensembles for cost-sensitive bankruptcy prediction. The selection of ensemble members is based on individual performance and pairwise diversity of classifiers. The experimental results on a real world database of French companies show that by selecting appropriate base classifiers the ensemble learning substantially improves the performance of cost-sensitive bankruptcy prediction.

Keywords: bankruptcy prediction, cost-sensitive classification, consensus decision making, classifier ensemble.

1 Introduction

Bankruptcy prediction is an extremely important research topic in financial decision-making with the recent emphasis on more powerful predictive models and better understanding of the financial data. A variety of approaches have been successfully applied using the traditional statistics and recently developed machine learning techniques. Bankruptcy prediction is formally solved as a classification task, first extracting the intrinsic patterns between the class (financial status) of companies and the features (financial and nonfinancial ratios) from the training samples, then predicting the future of new samples based on the derived models. It is well known that the costs of two misclassification errors, namely, classifying a bankrupt company as normal and classifying a normal company as bankrupt, are quite different. That is to say the misclassification cost is deemed to be a key factor in financial decision making in order to achieve accurate and desirable results. Cost-sensitive classification addresses the challenging problem, that is different (or asymmetric) costs are associated with different misclassification errors.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 266–276, 2013.

Many researchers have provided supporting evidences that the performance of predictive models can be significantly enhanced by means of ensemble computing methodology. A comprehensive review was presented in [15]. An ensemble is constructed by combining multiple classifiers which are trained individually and aggregated in a consensus manner when classifying new data. The rational behind ensemble design is to comprise appropriate base classifiers which have relatively high performance individually and low intercorrelation so as to ensure the effectiveness of the ensemble. Normally the ensemble members can be developed using: 1) different data sets as input to the same predictive model; 2) different subsets of features; 3) different predictive models implemented by non-hybrid structures (varying the parameters of classifiers) or hybrid structures (varying the type of classifiers). A variety of strategies commonly used to build an ensemble of classifiers include bagging, boosting, stacking, random subspace, rotation forest, class switching etc.

In the context of financial risk analysis, the ensemble methods emerged in recent years to boost the performance of single predictive model. It is encouraging that the ensemble with some simple (and weak) classifiers can produce prominent prediction results [13]. Nanni and Lumini tested various ensemble strategies using 4 diversified classifiers on 3 financial data sets. They found random subspace produced the best performance in terms of the area under ROC curve (AUC) [9]. Li and Sun used Case-based Reasoning (CBR) as the base classifier to construct an ensemble model based on diversified feature selection methods and weighted majority voting principle. The results confirmed the ensemble evidently improved the accuracy and stability of prediction [6]. On the other hand, some studies found the ensemble was not superior to the best single classifier consistently. The behaviour of base classifiers tended to vary across different ensemble strategies as shown by Marques et al. through the comparison of 5 ensemble methods and 7 base classifiers on 6 financial data sets [7]. Tsai and Wu also showed that an ensemble of diversified Neural Networks (NNs) did not always outperform the (single) best NN [14]. Finlay summarized the multiple classifiers systems into three categories: static parallel systems, multi-stage systems, and dynamic classifier systems. The empirical results of different multiple classifier systems demonstrated some delivered significantly better performance than the single best classifier, but many did not [2]. From this point of view, how to select the appropriate base classifiers and ensemble strategy remains a critical problem in the research of classifier ensemble.

In ensemble learning the multicollinearity problem indicates that the correlation are high among classifiers. It becomes more serious to stable learning methods such as Support Vector Machines (SVMs) in the sense that they are less sensitive to the change of training samples. The fundamental need for successful ensemble is therefore to comprise the base classifiers which have high (or moderate) performance individually and somewhat diversity that leads to independent misclassifications. Wang and Ma developed a hybrid ensemble model using bagging (for different training samples) and random subspace (for different features) strategies to ensure the diversity of multiple SVMs [16]. Sun and Li

selected the candidate classifiers by training an SVM using different kernel functions and feature subsets with the consideration of both individual performance and diversity analysis, and found the resulting SVM ensemble was superior to individual SVM [12]. Kim and Kang [4] have proposed a GA-based coverage optimization algorithm to determine the (near) optimal subsets of base classifiers with respect to prediction accuracy and variance influence factor (VIF). They demonstrated by selecting appropriate base SVMs the ensemble can produce prominent performance improvement.

Recent attempts are devoted to aggregating classifiers of different architectures to build an ensemble. Ravi et al. integrated MLP, RBF, PNN, SVM, CART, fuzzy rule-based classifier, PCA-MLP, PCARBF and PCA-PNN in a selective ensemble to reduce the prediction error rate [10]. Sun and Li proposed an ensemble method which combined MDA, Logit, NN, SVM, Decision tree, and CBR, with the conclusion that the ensemble indeed improved the prediction performance [11]. In another study, Huang and Chen aggregated Decision tree, MLP and SVM to enhance the predictive ability of the stand-alone classifiers [3]. The previous studies provided the evidences that a well designed hybrid ensemble can inherit advantages and avoid disadvantages of the employed methods, and thus outperform stand-alone classifiers. Canuto et al. investigated the relationship between the choice of ensemble members and accuracy through an extensive evaluation using different data sets and combination methods. A general conclusion was that the hybrid ensemble structures outperformed the non-hybrid ensemble structures in the accuracy consistently, owing to the higher diversity among ensemble members in different model types [1].

In summary, the prior studies demonstrate the ensemble learning is actually beneficial to improve the stand-alone classifiers though it is not always reported superior to the single best classifier. Yet rarely the misclassification cost, an issue of critical importance to bankruptcy decision making, has been considered in ensemble learning. In this paper we seek to study the performance of classifier ensembles by combining diversified cost-sensitive base classifiers. The base classifiers are implemented by different types of learning methods with identical representation of data. The ensemble is then built to comprise the classifiers which have low expected misclassification cost and somewhat diversity in output. The experimental results demonstrate the ensemble of properly selected base classifiers is effective to achieve superior and stable prediction.

The rest of the paper is organized as follows. Section 2 describes the data, classification methods, and classifier selection strategy. In Section 3 we show the experimental results of classifier ensembles and discuss how the choice of base classifiers affects the ensemble performance. Finally, in Section 4 we present the conclusions and point out the further lines of work.

2 Experiment Design

2.1 Data Description

Diane database contains financial statements of French companies. The problem goal is to find a model able to predict the class (healthy or bankrupt) of companies in a correct manner. After pre-processing the bankruptcy data set contains 1200 companies, in which 600 examples distressed in 2007, and the remainder are healthy. Table 1 lists the 30 financial ratios describing the features of companies in 2006. The data is normalized to unit range before feeding to the classifiers.

Table 1. Financial ratios of French Diane database

Variable Description	
x_1- Number of Employees Previous year	x_{16}- Cashflow / Turnover
x_2- Capital Employed / Fixed Assets	x_{17}- Working Capital / Turnover days
x_3- Financial Debt / Capital Employed	x_{18}- Net Current Assets/Turnover days
x_4- Depreciation of Tangible Assets	x_{19}- Working Capital Needs / Turnover
x_5- Working Capital / Current Assets	x_{20}- Export
x_6- Current ratio	x_{21}- Added Value per Employee in k EUR
x_7- Liquidity Ratio	x_{22}- Total Assets Turnover
x_8- Stock Turnover days	x_{23}- Operating Profit Margin
x_9- Collection Period days	x_{24}- Net Profit Margin
x_{10}- Credit Period days	x_{25}- Added Value Margin
x_{11}- Turnover per Employee k EUR	x_{26}- Part of Employees
x_{12}- Interest / Turnover	x_{27}- Return on Capital Employed
x_{13}- Debt Period days	x_{28}- Return on Total Assets
x_{14}- Financial Debt / Equity	x_{29}- EBIT Margin
x_{15}- Financial Debt / Cashflow	x_{30} - EBITDA Margin

Acronyms: EBIT: Earnings before interest and tax, EBITDA: Earnings before interest, tax, depreciation and amortization

2.2 Classifier Description

The bankruptcy prediction problem is extensively studied in the literature by various statistical and machine learning methods. We choose 10 methods in the experiments for a number of reasons. First, they utilize different learning methodologies. Second, they have received considerable academic attention in recent years. Third, they have been practically applicable in real world decision making problems. The default setting of all algorithms are used in Weka [17] without the consideration of parameter optimization.

- K-Nearest Neighbor (KNN) is an instance-based learning algorithm classifying an unknown instance to its nearest neighbors in the training data based on a specified distance metric. In the present work, we set the parameter K = 5 as recommended by previous studies.

- Multi-Level Perceptron (MLP) is a widely used artificial neural network for supervised learning. It consists of multiple layers of nodes, with each layer fully connected to the next one. The training of the network is to adjust the connection weights by means of back-propagation so to minimize the error between the network output and the desired output.
- Support Vector Machines (SVMs) are maximum margin classifiers in the sense that they find an optimal separating hyperplane which maximizes the margin between two classes of data in the kernel induced feature space. SVMs use the structural risk minimization principle to avoid overfitting. The polynomial kernel function is used in the experiments.
- Naive Bayesian Network (Bayes) estimates the probability of each class based on the assumption of feature independence.
- Bayesian logistic regression (BaysLR) uses a Laplace prior to avoid overfitting and produces sparse predictive models.
- J4.8 is the most widely used decision tree algorithm which first infers a tree-shape decision structure well-adapted to the training data then prunes the tree to avoid over-fitting.
- ADTree is an alternating decision tree for classification, composed of decision nodes and prediction nodes.
- RBF network implements a normalized Gaussian radial basis function network.
- Logistic Regression (LR) can be regarded as a generalized linear model used for binomial regression. It predicts the probability of financial distress by fitting data to a logistic function of the explanatory input variables.
- Decision Table (DecT) is a rule-based learning algorithm using a simple decision table majority classifier.

The cost-sensitive prediction is implemented by the meta Costsensitive Classifier in Weka. It is a general approach to make the base classifier cost-sensitive by assigning the weight to each training sample according to the predefined cost. In this way the aforementioned classifiers are able to handle the misclassification costs as a parameter.

2.3 Classifier Selection Strategy

We intend to select the classifiers with high (or moderate) performance and diversity in order to attain the effectiveness of an ensemble. To achieve this, two criteria are used for classifier selection, namely, expected misclassification cost (as the performance measure) and Q-statistic (as the diversity measure).

As a binary classification problem, bankruptcy prediction models produce a two dimensional confusion matrix containing the distribution of instances in the real class and predicted class. In the confusion matrix shown in Table 2, fn denotes the misclassification errors of 'bankrupt' companies as 'healthy', fp denotes the misclassification errors of 'healthy' companies as 'bankrupt', tp denotes the correct predictions of bankrupt samples, and tn denotes the correction predictions of healthy samples.

Table 2. Confusion matrix

real	predicted class	
class	bankrupt	healthy
bankrupt	tp	fn
healthy	fp	tn

Due to the asymmetric costs, the performance of predictive models should be evaluated by cost-relevant measures. We use expected misclassified cost (EMC) for the performance assessment. Assuming that C_p denotes the misclassification cost of a 'bankrupt' company, C_n denotes the misclassification cost of a 'healthy' company, and N denotes the total number of samples, EMC can be defined as:

$$EMC = C_p * \frac{fn}{N} + C_n * \frac{fp}{N} \tag{1}$$

When equal costs are used (i.e., $C_p = C_n$), EMC value is same to the traditional error rate $(fp + fn)/N$. For simplicity we set C_n as 1 and $C_p > 1$ as a real number. The cost ratio consequently denotes the relative importance of false negative errors with respect to false positive errors.

In the literature, the diversity of an ensemble can be measured by outcome diversity and structure diversity. The former evaluates how the classifiers are different in the outcome, and the latter evaluates how the classifiers are varied in the structure. Musehane et al. used Shannon-Wiener and Simpson to measure the diversity among multiple structurally different NNs [8]. The commonly used outcome diversity measures include pairwise measures (e.g., Q-statistic, correlation coefficient, disagreement, double fault) and non-pairwise measures (e.g., entropy measure, Kohavi-Wolpert variance, interrater agreement, difficulty index, generalized diversity, and coincident failure) [5]. In the present study, we use Q-statistic, a simple definition proposed by Yule [18] to measure the diversity between two classifiers M_i and M_j. In the following definition, a is the number of samples correctly classified by both classifiers, b is the number of samples incorrectly classified by both classifiers, c is the number of samples correctly classified by M_i but misclassified by M_j, and d is the number of samples correctly classified by M_j but misclassified by M_i. As a consequence, the value 1 of Q indicates the positive correlation between M_i and M_j (in other words, they tend to make decisions consistently), and the value -1 indicates the negative correlation (in other words, they tend to commit errors on different samples).

$$Q = \frac{ab - cd}{ab + cd} \tag{2}$$

By combining the above two criteria, we intend to select the candidate classifiers which have small EMC values individually and low pairwise Q-statistic values. An ensemble is then built on the selected base classifiers to achieve the consensus

decision. Out of various combination mechanisms, we use simple majority voting, that is to say the outcome of the ensemble is the class mostly predicted by the members.

To summarize, Figure 1 illustrates the framework of the consensus approach for cost-sensitive bankruptcy prediction, composed of model learning, classifier selection, and ensemble aggregation. The final decision is made with a consensus of the selected base classifiers.

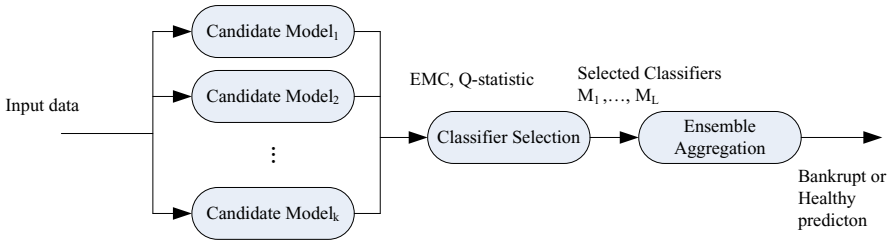


Fig. 1. Consensus approach for cost-sensitive bankruptcy prediction

3 Experimental Results

The experiments are performed on Diane database using the aforementioned 10 classifiers and the ensemble approach. For each specification of cost ratio, results are obtained by 10-fold cross validation. Table 3 shows the EMC values of the individual classifiers and ensembles at the varying cost ratio from 1 to 20. The values in bold indicate that the corresponding classifiers are selected to build the ensemble as described above. For each ensemble, Q_{av} is calculated as the averaged Q-statistic over all pairs of base classifiers.

Table 3. EMC results of individual classifiers and ensembles

Cost ratio	1	5	10	15	20
J48	**0.1092**	**0.2558**	**0.3158**	**0.3733**	**0.4050**
DecT	0.1275	0.3192	**0.4108**	0.5075	0.5000
KNN	0.1067	**0.2850**	**0.3933**	**0.5017**	**0.6100**
SVM	0.0908	0.2117	0.3050	0.3650	0.4250
MLP	0.0875	0.3158	0.4333	0.5008	0.5425
LR	**0.0875**	**0.1933**	**0.2200**	**0.2450**	**0.2683**
RBF	**0.1242**	**0.3325**	**0.4342**	**0.4850**	**0.5117**
Bays	0.1333	0.5358	**1.0100**	**1.4950**	**1.9467**
BaysLR	0.1208	0.3700	0.4850	0.5050	0.4975
ADTree	**0.1125**	0.2783	0.2883	0.3250	**0.3517**
Ensemble	0.0799	0.1783	0.2175	0.2308	0.2150
Q_{av}	0.8637	0.7724	0.6153	0.5502	0.6239

As was shown in this table, Logistic Regression always produces the lowest EMC value across different cost settings, thus it is selected to build the ensemble firstly. The other members are then selected successively in such a manner that they hold a relatively low EMC value and a low correlation with the preselected members. Here the threshold δ denotes the maximal correlation between two base classifiers. In this way, the size of ensemble is determined by the threshold. The experimental results show that the ensemble approach consistently produces a better performance than a single classifier in terms of EMC values, irrespective of the cost setting. It confirms that ensemble approach by selecting appropriate base classifies is beneficial to establish more accurate prediction models for the cost-sensitive bankruptcy prediction problem.

For further discussion we then take as an example the case when the cost ratio is 5 (i.e., the cost of classifying a bankrupt sample as healthy is 5 times more than classifying a healthy sample as bankrupt). In Figure 2, the performance of different ensembles is outlined at a varying correlation threshold δ. It is observed that the selection of members is important to the performance of ensembles. The best performance is achieved when four classifiers, namely LR, J48, KNN, and RBF, are selected at the threshold $\delta = 0.84$. In both cases, less or more candidates, there occurs a degradation of the ensemble performance. The selected classifiers are shown in Table 4 given different thresholds. It is interesting that some strong classifiers such as SVM, despite of good performance individually, are not selected by the ensemble due to the high correlation with others. On the contrary, some weak classifiers such as KNN and RBF network are selected owing to the moderate performance and relatively high diversity with the strong classifiers which have been selected. In this case, RBF achieves a moderate EMC (0.3325), but holds a lowest Q-statistic with the other classifiers in the ensemble (0.7334 on the average and 0.7458 on the maximum) as shown in Figure 3. The results give some insights into the choice of base classifiers when building the ensemble.

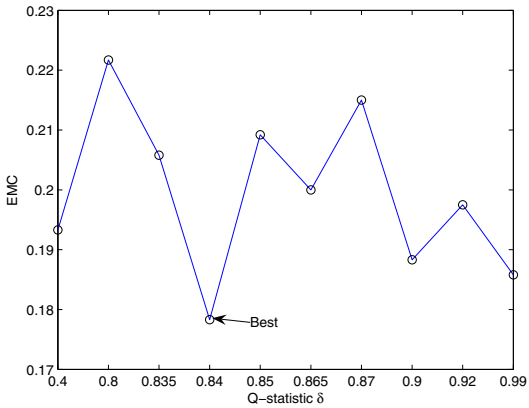


Fig. 2. Performance of classifier ensembles at varying δ (cost ratio = 5)

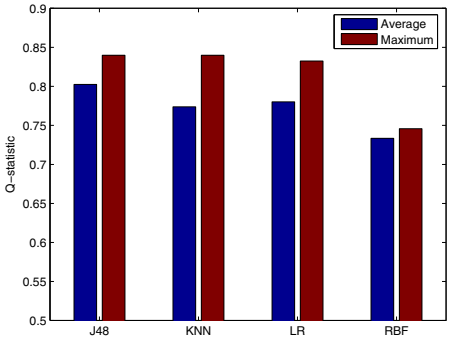


Fig. 3. Averaged and maximal Q-statistic value of selected classifiers (cost ratio = 5, $\delta = 0.84$)

Table 4. Ensembles at different δ (cost = 5)

δ	Selected base classifiers	EMC
0.4	LR	0.1933
0.8	LR, RBF	0.2217
0.835	LR, KNN, RBF	0.2058
0.84	**LR, J48, KNN, RBF**	**0.1783**
0.85	LR, J48, KNN, DecT, RBF	0.2092
0.865	LR, J48, KNN, DecT, RBF, BaysLR	0.2000
0.87	LR, J48, KNN, DecT, RBF, BaysLR, Bays	0.2150
0.9	LR, J48, ADTree, KNN, DecT, RBF, BaysLR, Bays	0.1883
0.92	LR, J48, ADTree, KNN, MLP, DecT, RBF, BaysLR, Bays	0.1975
0.99	LR, SVM, J48, ADTree, KNN, MLP, DecT, RBF, BaysLR, Bays	0.1858

4 Conclusion and Future Work

The combination of multiple classifiers has emerged as an effective approach to improve the single classifier. It has the general aim to improve classification results by using several classifiers and also to use them jointly more intelligently than just as individual classifiers. The previous studies have demonstrated the benefit of ensemble learning in the prediction of business failure. To our knowledge few papers studied the ensemble of cost-sensitive classifiers for bankruptcy prediction. In this paper we investigated the performance of classifiers ensembles with properly selected members in the context of cost-sensitive bankruptcy prediction. The ensemble is constructed based on the individual performance and pairwise diversity of candidate classifiers. The experiments on a French database show that a properly constructed ensemble yields improved prediction performance over a single classifier. Some conclusions can be attained. 1) The ensemble technique actually contributes to boosting the performance of stand-alone cost-sensitive classifiers for this problem. 2) The choice of base classifiers is a key factor in designing classifier ensembles, where the individual performance and

outcome diversity of classifiers are two helpful criteria. 3) Although weak classifiers yield only moderate performance individually, they are potential to increase the overall performance of an ensemble. It is suggested to combine strong classifiers with weak classifiers of high outcome diversity in an ensemble to ensure the effective fusion of classifiers.

In the future work, some limitations will be addressed. First, although the classifier selection approach is viable, the proper value of correlation threshold is determined by a trail and error approach with the known outcome and thus might yield a local optimal solution. A further step is using global search to build the committee which achieves the optimal performance. Second, an extensive evaluation will be conducted to validate the generalizability of the results using other data sets, cost-sensitive methods and learning models. Last, there are a wide range of strategies to generate diverse classifiers for building the ensemble, for example by varying the training samples, features, or parameters of classifiers. Besides, the combination method is also a component which affects the performance of ensembles. It is interesting to compare the performance of different ensemble approaches in the future study.

Acknowledgements. This work was partly supported by Natural Science Foundation of China with Grant No. 91024004.

References

1. Canuto, A.M.P., Abreu, M.C.C., de Melo Oliveira, L., Xavier Jr., J.C., de, A., Santos, M.: Investigating the influence of the choice of the ensemble members in accuracy and diversity of selection-based and fusion-based methods for ensembles. Pattern Recognition Letters 28(4), 472–486 (2007)
2. Finlay, S.: Multiple classifier architectures and their application to credit risk assessment. European Journal of Operational Research 210(2), 368–378 (2011)
3. Hung, C., Chen, J.-H.: A selective ensemble based on expected probabilities for bankruptcy prediction. Expert Systems with Applications 36(3, pt. 1), 5297–5303 (2009)
4. Kim, M.-J., Kang, D.-K.: Classifiers selection in ensembles using genetic algorithms for bankruptcy prediction. Expert Systems with Applications 39(10), 9308–9314 (2012)
5. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning 51(2), 181–207 (2003)
6. Li, H., Sun, J.: Principal component case-based reasoning ensemble for business failure prediction. Information & Management 48(6), 220–227 (2011)
7. Marques, A.I., Garcia, V., Sanchez, J.S.: Exploring the behaviour of base classifiers in credit scoring ensembles. Expert Systems with Applications 39(11), 10244–10250 (2012)
8. Musehane, R., Netshiongolwe, F., Nelwamondo, F.V., Masisi, L., Marwala, T.: Relationship between diversity and perfomance of multiple classifiers for decision support. Computing Research Repository, abs/0810.3 (2008)

9. Nanni, L., Lumini, A.: An experimental comparison of ensemble of classifiers for bankruptcy prediction and credit scoring. Expert Systems with Applications 36(2, pt. 2), 3028–3033 (2009)
10. Ravi, V., Kurniawan, H.: Peter Nwee Kok Thai, and P. Ravi Kumar. Soft computing system for bank performance prediction. Applied Soft Computing 8(1), 305–315 (2008)
11. Sun, J., Li, H.: Listed companies' financial distress prediction based on weighted majority voting combination of multiple classifiers. Expert Systems with Applications 35(3), 818–827 (2008)
12. Sun, J., Li, H.: Financial distress prediction using support vector machines: Ensemble vs. individual. Applied Soft Computing 12(8), 2254–2265 (2012)
13. Sun, J., Jia, M.Y., Li, H.: Adaboost ensemble for financial distress prediction: An empirical comparison with data from Chinese listed companies. Expert Systems with Applications 38(8), 9305–9312 (2011)
14. Tsai, C.-F., Wu, J.-W.: Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Systems with Applications 34(4), 2639–2649 (2008)
15. Verikas, A., Kalsyte, Z., Bacauskiene, M., Gelzinis, A.: Hybrid and ensemble-based soft computing techniques in bankruptcy prediction: A survey. Soft Computing - A Fusion of Foundations, Methodologies and Applications 14(9), 995–1010 (2010)
16. Wang, G., Ma, J.: A hybrid ensemble approach for enterprise credit risk assessment based on support vector machine. Expert Systems with Applications 39(5), 5325–5331 (2012)
17. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical machine learning tools and techniques, 3rd edn. Morgan Kaufmann, San Francisco (2011)
18. Yule, G.: On the association of attributes in statistics. Philosophical Transactions of the Royal Society of London, Series A 194, 257–319 (1900)

On the Regularization Parameter Selection for Sparse Code Learning in Electrical Source Separation

Marisa Figueiredo, Bernardete Ribeiro, and Ana Maria de Almeida

CISUC, Department of Informatics Engineering, University of Coimbra
{mbfig,bribeiro,amaria}@dei.uc.pt

Abstract. Source separation of whole-home electrical consumption also known as energy disaggregation plays a crucial role in energy savings and sustainable development. One important approach towards accurate energy disaggregation is based on sparse code learning. The sparsity-based source separation algorithms allow to build models that explicitly generalize across multiple different devices of the same category. While this method has recently been investigated, yet the importance of the degree of sparseness given by the regularization parameter is rarely considered. In this paper we aim at investigating the performance of learning representations from the aggregated electrical load signal with sparse models for energy disaggregation. In particular we focus our study on the influence of the regularization parameter in the overall approach. The computational experiments yielded in real data from home electrical energy consumption show that for several degrees of sparseness a reliable scheme for energy disaggregation can be obtained with statistical significance.

Keywords: Sparse Coding Learning, Electrical Signal Disaggregation, NILM, Parameter Regularization.

1 Introduction

Energy disaggregation via sparse coding for non-intrusive (appliance) load monitoring, aka NILM or NIALM [1,2], was proposed by Kolter *et al.* in [3]. The goal of NILM is to separate the whole-home energy consumption (aggregated signal) into the individual signals of each appliance in the electrical network (see Fig. 1), the problem of disaggregation can be viewed as a source separation one. NILM related research has advanced in the recent years following approaches based on appliances' signatures [4,5,6] as initially proposed by Hart. However, methods that learn data-adaptive representations usually applied to source separation problems, as sparse coding and non-negative matrix factorization (NMF), are also suitable. In fact, [3] proposes a sparse coding method for electrical disaggregation. This means that, sparse representations of electrical consumption for each device in the network are learned, which are then enriched with information supplied by the whole-home signal. Then, disaggregation is obtained for a set of

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 277–286, 2013.

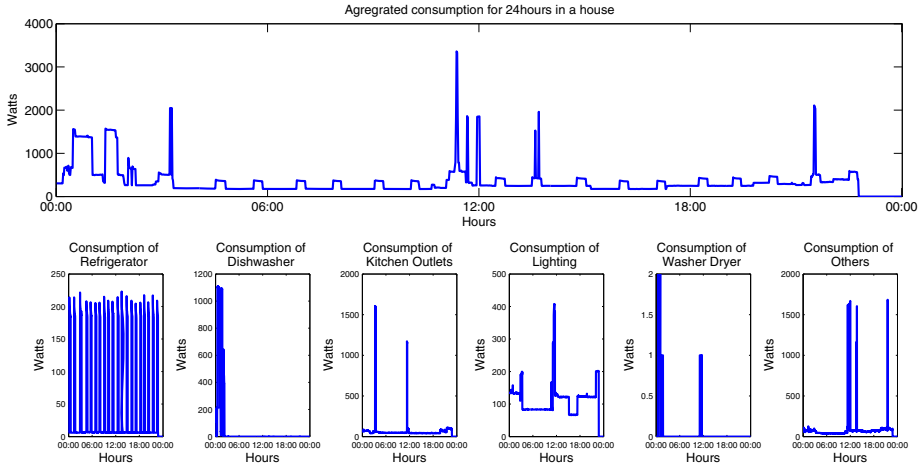


Fig. 1. The home aggregated (and device) consumption signals for 24hours' period

unknown aggregated signals. Since the electrical consumption is always a non-negative quantity, this approach imposes non-negative restrictions in addition to the sparsity condition.

In this context, while the non-negativity restrictions are clear, the imposition of sparsity conditions is not obvious in regard of the signal in analysis. The aggregated signal represents the consumption of two classes of appliances: the ones that are switch on and off by the household members and the ones that once they are on they operate automatically. The former may include devices with stand-by mode, representing a low consumption, and the latter may also have a residual consumption in parts of the operation cycle (see the Refrigerator graphic in Fig. 1). Thus, the associated sparseness degree of these consumptions would never be very high. Consequently, an accurate representation would correspond to lower values for the parameter in the sparse code learning process. In this work, it is named of regularization parameter to be consistent with [3].

Since the real contribution of the sparsity condition is still unclear for energy disaggregation, this paper presents a study of the true relevance of this parameter. Towards this end, a computational experience was outlined using real world data in order to evaluate the influence of this condition in an accurate disaggregation. Therefore, representations were learned by the approach proposed in [3] considering several degrees of sparseness and statistical tests were used to validate the conclusions.

The remainder of this paper is organized as follows. Next section introduces the necessary background, followed by a discussion about the sparsity condition considered by the approach under study in Section 3. Section 4 presents the experimental setup, results and correspondent discussion. Conclusions and directions of future work are given in Section 5.

2 Preliminaries

The goal of a source separation problem is to recover the original source signals from the several mixed signals sensed. It is assumed that some knowledge about the sources or the mixing process is available. In the following we considered that the mixed (aggregated) and source (device) signals are given for the initial training phase whereas at the testing phase only the mixed signals are known.

Given a signal $\bar{x} = [\bar{x}(1), \bar{x}(2), \ldots, \bar{x}(T)]^T$ corresponding to the aggregated electrical consumption during a period of time T, the goal is to recover the k signals $x_i = [x_i(1), x_i(2), \ldots, x_i(T)]^T, i = 1, \ldots, k$ associated to each device $i = 1, \ldots, k$. Moreover, we consider that $\bar{x} = \sum_{i=1}^{k} x_i$. Similarly, given a set of m aggregated signals and the correspondent m device signals, each column of $\bar{X} \in \mathbb{R}^{T \times m}$ is the m-th aggregated signal and likewise each column of $X_i \in \mathbb{R}^{T \times m}$ is the m-th signal for device i. Again, $\bar{X} = \sum_{i=1}^{k} X_i$.

To solve this source separation problem, a model is trained considering that matrices $\bar{X}$ and X_i are available. At the test step, the only accessible data is a new set of m' aggregated signals, forming $\bar{X}' \in \mathbb{R}^{T \times m'}$ where the goal is to obtain its decomposition into $X_i', i = 1, \ldots, k$, that is, the signals associated to each source. Regarding that energy consumption is always a non-negative quantity, matrices $\bar{X}$, $\bar{X}'$, $X_i, i = 1, \ldots, k$ and X_i' are non-negative. Hence, non-negative approaches are more suitable to solve the task at hand.

3 Sparsity Modeling Approach for Energy Disaggregation

3.1 Sparse Condition

In a sparse model, a given signal $x \in \mathbb{R}^T$ is represented by a set of r basis vectors $B \in \mathbb{R}^{T \times r}$, known as dictionary, and by a sparse vector $a \in \mathbb{R}^r$ (sparse code) such that $x \approx Ba$. The objective function of this decomposition is

$$\min_{a \in \mathbb{R}^r} \frac{1}{2} \|x - Ba\|_2^2 + \lambda \phi(a)$$

where $\lambda \phi(a)$ is a sparsity inducing regularization for a, which represents a trade-off between sparseness and accurate reconstruction [7]. Usually, a sparse coding framework for source separation consists in training separate models for each class i, which are then concatenated and used in the test step in order to separate the aggregated signal. Formally, we start by defining sparse models for device i such that $X_i \approx B_i A_i$, where the columns of $B_i \in \mathbb{R}^{T \times r}$ represent r basis functions and the columns of $A_i \in \mathbb{R}^{r \times m}$ consist in the activations (sparse codes) of this dictionary. Within the electrical disaggregation context, and as explained earlier, a restriction for non-negativity of the bases and activations matrices is added.

Sparse coding is an interesting solution for energy disaggregation but still it demands the definition of a degree of sparseness. Observing the data in analysis (Fig. 1) we deduced that the sparsity in X_i and $\bar{X}$ is low, with exception

of particular devices. Thereby also the λ associated to an accurate disaggregation would be low. Consequently, the imposition of sparseness would be of little significance for the learning process of each model, apart from being time consuming [3]. Moreover, approaches that learn representations based on the data enforce the sparseness by itself. For instance, "the non-negativity constraint by itself already enforces some sparseness on the resulting representation of data when using NMF" [8]. A NMF algorithm can not control the sparsity included. In order to support these observations, we tested the method by exploring the importance of the value of λ for enforcing sparseness in the learning process.

3.2 Discriminative Disaggregation Sparse Coding

The Discriminative Disaggregation Sparse Coding (DDSC) [3], is a sparse coding based approach for energy disaggregation. The DDSC is described in Algorithm 1 and has three main steps: i) sparse coding pre-training, ii) discriminative disaggregation training and iii) test phase. Firstly, a sparse model for each device

Data: $X_i, i = 1, \ldots, k,\ \bar{X}, \bar{X}',\ \lambda \in \mathbb{R}_+,\ \alpha \in \mathbb{R}_+\ r \in \mathcal{N}, e \in \mathbb{R}_+$
Result: $\hat{X}'_1, \ldots, \hat{X}'_k \in \mathbb{R}^{T \times m'}$
`/* Pre-training:` */
Initialize B_i and A_i with positive values ;
Columns of B_i must have unit norm;
for *each* $i = 1, \ldots, k$ **do**
  while $|$*Objective value of iteration* j − *Objective value of iteration* $j-1| > e$ **do**
    $A_i \leftarrow \arg\min_{A_i \geq 0} \|X_i - B_i A\|_F^2 + \lambda \sum_{p,q} A_{p,q}$;
    $B_i \leftarrow \arg\min_{B_i \geq 0} \|X_i - B A_i\|_F^2$;
    Objective value of iteration $j = \frac{1}{2} \|X_i - B_i A_i\|_F^2 + \lambda \sum_{p,q} A_i\{p,q\}$
  end
end
`/* Discriminative disaggregation training:` */
Set $A^*_{1:k} \leftarrow A_{1:k}, \tilde{B}_{1:k} \leftarrow B_{1:k}$;
while $|$*Objective value of iteration* j − *Objective value of iteration* $j-1| > e$ **do**
  $\hat{A}_{1:k} \leftarrow \arg\min_{A_{1:k}} F\left(\bar{X}, \tilde{B}_{1:k}, A_{1:k}\right)$;
  $\tilde{B} \leftarrow \left[\tilde{B} - \alpha\left(\left(\bar{X} - \tilde{B}\hat{A}\right)\hat{A}^T - \left(\bar{X} - \tilde{B}A^*\right)A^{*T}\right)\right]_+$;
  for *all the columns of* $\tilde{B}$, $\tilde{b}_i^{(j)}$ **do**
    $\tilde{b}_i^{(j)} \leftarrow \frac{\tilde{b}_i^{(j)}}{\left\|\tilde{b}_i^{(j)}\right\|_2}$
  end
  Objective value of iteration $j = \frac{1}{2}\left\|\bar{X} - \tilde{B}\hat{A}\right\|_F^2 + \lambda \sum_{p,q} \hat{A}_{p,q}$
end
`/* Test:` */
$\hat{A}'_{1:k} \leftarrow \arg\min_{A_{1:k}} F\left(\bar{X}', \tilde{B}_{1:k}, A_{1:k}\right)$;
Predict $\hat{X}'_i = B_i \hat{A}'_i$;

Algorithm 1. The DDSC algorithm [3]

is computed using a coordinate descent approach [9,10] for the activations and the multiplicative NMF update proposed in [8] to obtain the optimization over B_i. Note that $\lambda \in \mathbb{R}_+$ is a regularization parameter, $\|\bullet\|_F$ is the Frobenius norm and $\|\bullet\|_2$ is the l_2 norm. Next, the discriminative disaggregation training incorporates data supplied by $\bar{X}$ in the bases $B_i, i = 1, \ldots, k$, considering

$$\hat{A}_{1:k} = \arg\min_{A_{1:k}} \left\| \bar{X} - [B_1 \dots B_k] \begin{bmatrix} A_1 \\ \vdots \\ A_k \end{bmatrix} \right\|_F^2 + \lambda \sum_{i,p,q} (A_i)_{p,q} \quad (1)$$
$$\equiv \arg\min_{A_{1:k} \geq 0} F\left(\bar{X}, B_{1:k}, A_{1:k}\right),$$

where $X_{1:k}$, $B_{1:k}$ and $A_{1:k}$ represent $X_1, \dots, X_k$, $B_1, \dots, B_k$ and $A_1, \dots, A_k$, respectively. $\hat{A}_1, \dots, \hat{A}_k$ are the activations associated to the aggregated signal. The best value for $\hat{A}_i$ is $A_i^* = A_i$. Thereby, $B_{1:k}$ is optimized such that $\hat{A}_{1:k}$ and $A_{1:k}^*$ are as close as possible. Moreover, the reconstruction bases $B_{1:k}$ learned in the pre-training, are replaced by the disaggregation bases $\tilde{B}_{1:k}$ in Equation 1. Finally, given a set of aggregated signals $\bar{X}'$ and the bases $\tilde{B}$, the activations $\hat{A}'_{1:k}$ are calculated as well as the signals of the ith device $\hat{X}'$. Further details can be found in [3].

4 Computational Experiments

4.1 Experimental Setup

REDD[1] consists of whole-home and circuit/device specific electricity consumption for a number of real houses over several months time. For each monitored house, the whole home electricity signal up to 24 individual circuits in the home, each labeled with its category of appliance were recorded [11]. A preprocessing phase selected only the common periods of sampling for both aggregated and individual signals. They were then downsampled using a median filter such that each sample became spaced 5 minutes apart. The data of similar equipments was added and, based on the percentage that each group represents in the total electricity consumed, the five ones presenting higher impact were selected. The sixth one contains the remaining data that was not measured by individual meter and also data of remaining equipments. The signals were subjected to a normalization step assuring that the relative importance of each group in the aggregated signal was maintained. The elements of each signal (aggregated and device ones) were normalised regarding the norm of the aggregated signal. Table 1 describes the post-processed dataset and the number of daily signals used for training and test, this is, the number of columns in $\bar{X}, X_i$ and $\bar{X}'$. For this experiment, the DDSC was implemented in Matlab, the maximum number of iterations was set to 1000 and the error value to 0.00001. Regarding our goal, we fixed α, one of the hyper-parameters of DDSC, to 0.0001 and a grid search was used to test several values for the number of bases r and for the degree of sparseness λ. In particular, we used values of $r \in \{10, 15, 20, 25, 30\}$ and $\lambda \in \{0.0001, 0.0005, 0.001, 0.005, 0.01\}$.

We evaluated the performance of the DDSC in terms of i) disaggregation error; ii) root-mean-square error (RMSE) for the aggregated signal and individual signals of the appliances; and iii) correlation between the estimated and true signals.

[1] `http://redd.csail.mit.edu`

Table 1. Post-processed REDD Dataset

		House 1	House 2	House 3	House 4	House 5	House 6
#Days		23	16	23	25	8	16
#Training		16	11	16	17	6	11
#Test		7	5	7	8	2	5
Groups	Refrigerator	✓	✓	✓		✓	✓
	Dishwasher	✓	✓	✓	✓		
	Kitchen outlets	✓	✓		✓		
	Lighting	✓	✓	✓	✓	✓	✓
	Washer Dryer	✓		✓			
	Microwave		✓				
	Electronics			✓		✓	
	Furnace				✓	✓	
	Stove				✓		
	Subplane					✓	
	Outlet unknown						✓
	Electric Heat						✓
	Air Conditioning						✓
	Others	✓	✓	✓	✓	✓	✓

An interesting measure would be the disaggregation error: $\sum_{i=1}^{k} \frac{1}{2} \left\| X_i - \hat{X}_i \right\|_F^2$ where X_i is the matrix of measured signals for equipment i and $\hat{X}_i$ its predicted version. We also compute the RMSE, which provides a measure of the error between the predicted values and the actually observed ones. In this work, for a general overview of the error, we used $RMSE(\bar{X}, \hat{\bar{X}}) = \sqrt{\frac{\sum_t^T \sum_d^m \left(\bar{X} - \hat{\bar{X}}\right)^2}{T * m}}$ where $\bar{X}$ is the aggregated signal and $\hat{\bar{X}}$ its predicted version. For a more detailed analysis we also calculated the correspondent $RMSE_i$ associated to each group $i = 1, \ldots, k$ of devices replacing $\bar{X}$ by X_i and $\hat{\bar{X}}$ by the prediction $\hat{X}_i$. For each device, the relation between each daily predicted signal and the measured consumption was accessed by the correlation coefficient. The reported values represent the mean correlation coefficient of the signals for each appliance.

4.2 Results and Discussion

The following results are the mean of 30 runs. Due to the small number of signals associated to the test set for House 5, the results were not used for this analysis. Also, regarding the similar results of House 1 and House 2, we report only those associated to House 1. We first look into the disaggregation error obtained by the DDSC for each of the 5 houses in study (Fig. 2). A decreasing trend is clear for the training results: as the r value increases, the disaggregation error decreases along the several λ values. For a fixed value for r value we can observe that the lower λ becomes, the lower the disaggregation errors is. For instance, in House 6 and with $r = 30$, the disaggregation error for $\lambda = 0.01$ is 4.20 times higher than the corresponded value for $\lambda = 0.0001$. For all the houses, $\lambda = 0.01$ has the highest disaggregation error for every r. Regarding the test results, a similar decreasing trend along the r is noticed for House 3 and 4. For the latter, $\lambda = 0.01$ corresponds to the highest disaggregation errors. Nevertheless, for House 6 with $r = 30$, the disaggregation error for $\lambda = 0.01$ represents 84.54% of the value

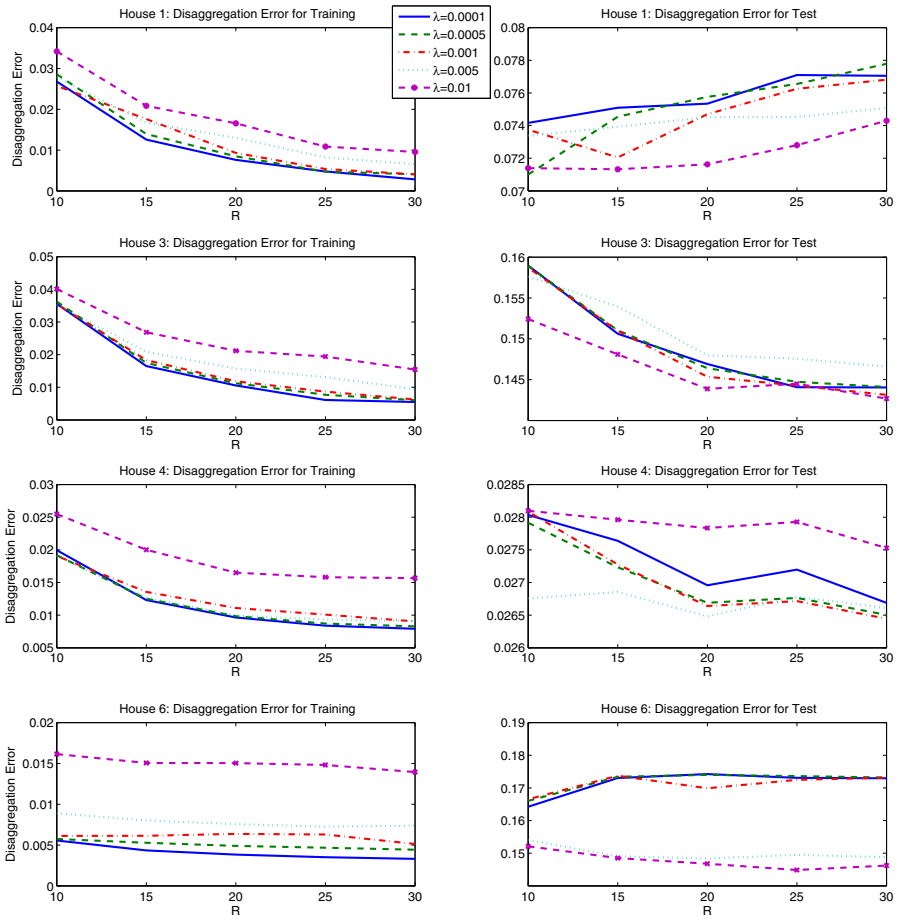


Fig. 2. The disaggregation error for each house in training and test

obtained for $\lambda = 0.0001$. Notice that these observations are similar to the ones that can be drawn for the remaining houses. Apart from House 4, which has similar ranges for the training and test results, the test disaggregation values range between $[0.07, 0.17]$ while the training ones range from 0 to 0.05.

We also investigated RMSE values for the aggregated signal (Fig. 3). Here the trend is also obvious: for all the houses, either in training or in test, the RMSE values decrease as r increases. Furthermore, the higher the value of λ, the higher is the associated RMSE value, with the exception of the test results in House 4. In general, RMSE values for $\lambda = 0.01$ and $\lambda = 0.005$ surpass the RMSE for the remaining λs. Again, apart from House 4 which has similar ranges for the training and test results, the RMSE values for test exceed by far the training ones, in particular for House 6. In this case, for $r = 30$ the RMSE associated to $\lambda = 0.01$ corresponds to 3.0227 times the RMSE of $\lambda = 0.0001$, for the training and 1.1450

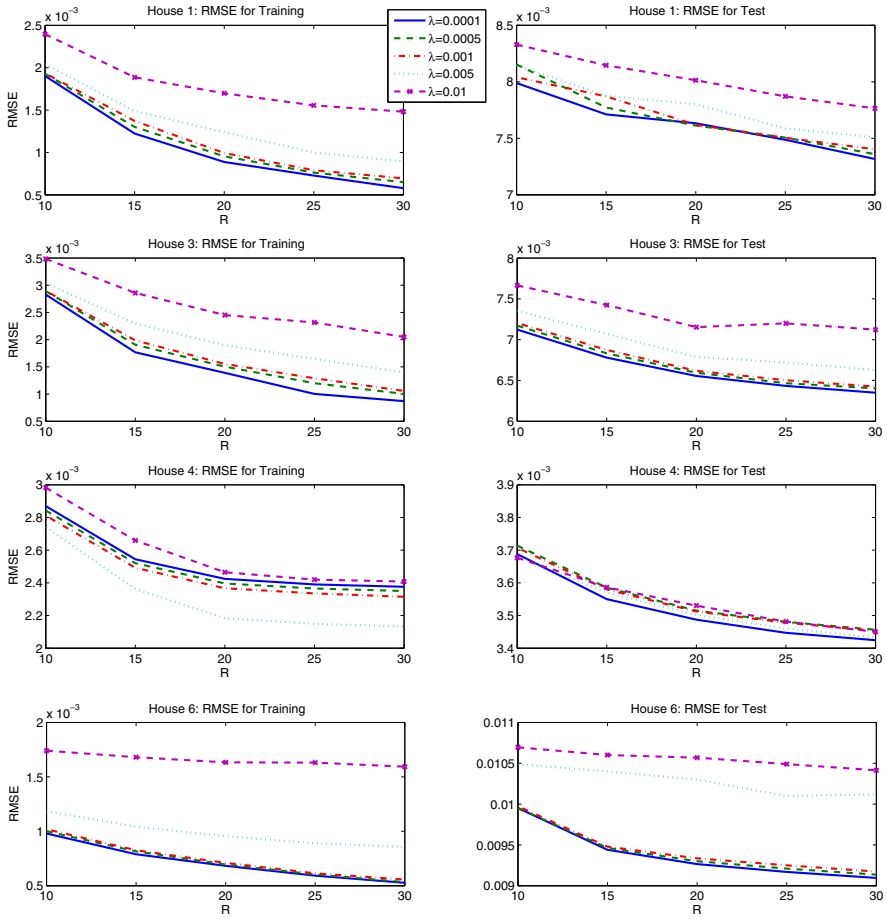


Fig. 3. The aggregated RMSE for each house in training and test

times for the test. For further details, we computed the RMSE by devices with $r = 30$. We noticed a common decreasing trend, in training, as the degree of sparseness decreases, in accordance with the observations of aggregated RMSE. Nevertheless, no significant decrease was observed for the test results. This particular analysis allows us to identify the groups poorly disaggregated: "Others", "Dishwasher" and "Washer Dryer" for House 1, "Lighting" and "Washer Dryer" for House 3, "Others" and "Lighting" for House 5 and "Air Conditioning" for House 6.

To clarify the true importance of a higher degree of sparseness, statistical tests were performed to analyse the existence of similarities between the RMSE values of all the houses in the dataset for $\lambda = 0.0001$ and $\lambda = 0.01$, considering $r = 30$. According to the Kolmogorov-Smirnov test at a significance level of 0.05, the RMSE values of each λ value for training and test are far from a normal

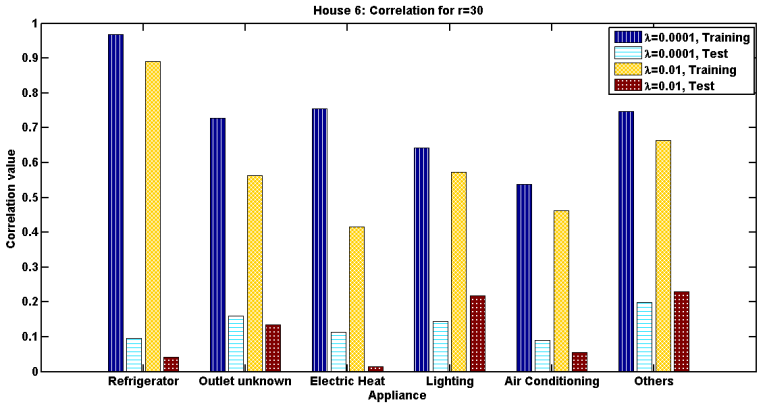


Fig. 4. The correlation coefficient for House 6 ($r = 30$, $\lambda = 0.0001$ and $\lambda = 0.01$)

distribution (all the $p - values < 0.001$). Therefore, a Mann-Whitney test was used. According to its results, the distribution associated to RMSE is not the same for both λs, neither in training nor in test at a significance level of 0.05. In fact, and regarding the mean values (Mdn), the RMSE values for $\lambda = 0.01$ ($Mdn = 0.0018$) are higher than the ones for $\lambda = 0.0001$ ($Mdn = 0.0006$) for the training, $U = 4077.00$, $z = -9.548$. Moreover, for the test, similar observations can be drawn: the RMSE values associated $\lambda = 0.01$ ($Mdn = 0.0077$) differ from the ones for $\lambda = 0.0001$ ($Mdn = 0.0074$), $U = 8202.00$, $z = -4.057$. To conclude, Fig. 4 reinforces the idea that, in general, forcing a higher sparsity degree does not imply better results for the energy disaggregation.

5 Conclusions

Recently sparse coding methods for energy disaggregation have successfully been proposed. Yet the adequacy of the sparse condition for the signals in analysis is not obvious. In fact, when using real-world data, periods of very low consumption are not very significant mainly due to stand-by consumptions and "always on" appliances. This study focus on the sparseness regularization parameter and its importance for successful electrical source separation using sparse coding. For this purpose, we analysed the results obtained by a method designed of energy disaggregation based on a sparse coding framework for source separation. Experiments used a dataset composed by residential electrical signals for which several degrees of sparseness were considered to learn a electrical representation for energy disaggregation. The results indicated that a statistical significant difference exists between the lower and the higher sparsity degree tested in terms of root-mean-square-error of the signals. Furthermore, in this context, the best results were obtained when imposing the lowest sparseness degrees. Even though

the presented analysis indicates that a representation learned by an approach without (or with very low) sparsity requirements can be better suited for energy disaggregation, we will proceed exploring this conjecture in future work, as well as a solution involving output-weight regularization.

Acknowledgments. FCT (Foundation for Science and Technology) is gratefully acknowledged for funding the first author with the grant SFRH/BD/68353/2010.

References

1. Hart, G.W.: Nonintrusive appliance load monitoring. Proc. of the IEEE 80, 1870–1891 (1992)
2. Sultanem, F.: Using appliance signatures for monitoring residential loads at meter panel level. IEEE Transactions on Power Delivery 6, 1380–1385 (1991)
3. Zico Kolter, J., Batra, S., Ng, A.: Energy disaggregation via discriminative sparse coding. In: Lafferty, J., Williams, C.K.I., Shawe-Taylor, J., Zemel, R., Culotta, A. (eds.) Advances in Neural Information Processing Systems, vol. 23, pp. 1153–1161 (2010)
4. Cole, A., Albicki, A.: Algorithm for non intrusive identification of residential appliances. In: Proc. of the IEEE Intl. Symposium on Circuits and Systems, vol. 3, pp. 338–341 (1998)
5. Patel, S.N., Robertson, T., Kientz, J.A., Reynolds, M.S., Abowd, G.D.: At the Flick of a Switch: Detecting and Classifying Unique Electrical Events on the Residential Power Line (Nominated for the Best Paper Award). In: Krumm, J., Abowd, G.D., Seneviratne, A., Strang, T. (eds.) UbiComp 2007. LNCS, vol. 4717, pp. 271–288. Springer, Heidelberg (2007)
6. Figueiredo, M., De Almeida, A., Ribeiro, B.: Home electrical signal disaggregation for non-intrusive load monitoring (NILM) systems. Neurocomputing 96, 66–73 (2012)
7. Hoyer, P.O.: Non-negative sparse coding. In: Proc. IEEE Workshop on Neural Networks for Signal Processing, pp. 557–565 (2002)
8. Eggert, J., Korner, E.: Sparse coding and nmf. In: Proc. IEEE Intl. Joint Conf. on Neural Networks, vol. 4, pp. 2529–2533 (2004)
9. Friedman, J., Hastie, T., Höfling, H., Tibshirani, R.: Pathwise coordinate optimization. The Annals of Applied Statistics 1(2), 302–332 (2007)
10. Friedman, J., Hastie, T., Tibshirani, R.: A note on the group lasso and a sparse group lasso. Technical report, Stanford University (2010)
11. Kolter, J.Z., Johnson, M.J.: REDD: A Public Data Set for Energy Disaggregation Research. In: Proc. of SustKDD Workshop on Data Mining Applications in Sustainability (2011)

Region Based Fuzzy Background Subtraction Using Choquet Integral

Muhammet Balcilar and A. Coskun Sonmez

Department of Computer Engineering, Yildiz Technical University, Istanbul, Turkey
{muhammet,acsonmez}@ce.yildiz.edu.tr

Abstract. Background subtraction, is a widely used method for identifying moving objects in multimedia applications such as video surveillance. Deterministic approaches are the first applications in literature, and they followed statistical approaches; however, more recently prediction-based filter approaches are preferred by researchers. The methods suggested for background subtraction in traffic surveillance applications, which are subject to many uncertainties, such as illumination noise, sudden changes in ambient light and structural changes, have to date failed to satisfy the requirements. Fuzzy approaches in the Artificial Intelligence method are widely used by researchers to eliminate uncertainties within the problem. In this study, a fuzzy background subtraction method, using choquet integral that process certain group of pixels together in order to eliminate uncertainties is suggested. The method is tested on traffic surveillance dataset, leading to satisfying results.

Keywords: Background Subtraction, Fuzzy Measure, Choquet Integral, Traffic Surveillance Camera.

1 Introduction

The background of a video can be defined as the static part on the scene that has no belong to any moving object being depicted. Once the background parts in a video are distinguished, it becomes much easier to perceive, recognize and classify moving objects. Even though the background is expressed as the static parts of a video, problems such as lighting conditions, camera oscillation, periodic movement, such as shaking leaves or the motion on the sea, become distorted, and structural changes in the background (e.g. road paving) make background subtraction more difficult. As backgrounds change always over time, their estimation is not an element of work that can be carried out once and then disregarded, as it must be repeated constantly over time. Background subtraction has been the subject to much research, and many approaches have been developed to date that can be divided among four groups [1].

- Deterministic Methods
- Statistical Methods
- Filter-Based Estimation Methods
- Fuzzy Methods

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 287–296, 2013.

The method proposed by Valestin and Cuchiara in [2, 3] is assigning as background the median and mean values of n number of past data for each pixel. This method requires a large amount of memory depending on the n value. In the running average method, past values are not stored in the memory, and new values affect the background in proportion with the learning coefficient, which reduces memory requirement. Pfinder algorithm named by Wren et al in [4] depends on the assumption that the background color value will fit normal distribution. By computing to what extent the new incoming color value is similar to the Gaussian distribution representing background color probability, based on this similarity measurement; the variance and mean values of the Gaussian distribution in question are updated. By improving this method, Stauffer and Grimson in [5] have developed several background methods. In this way, backgrounds with multiple situations such as wavy sea, moving leaves can model. The method proposed by Han et al in [6] is a mean-shift based background subtraction. This method calculates the probable background values through iterative operations by working on the histogram of n number of past color value at each pixel. In the method proposed by Karmann and Brant in [7], the time-varying value at each pixel is assumed to be an indicator, and what this indicator is supposed to be for the subsequent time step is estimated using the Kalman filter. Thus, the background is estimated for the subsequent frame. Proposed study in [8], kalman filter is used to extract the background from traffic surveillance camera.

Uncertainties may be eliminated when attempting to resolve problems through the use of fuzzy background subtraction systems, which consist of fuzzy rules [9]. Background subtraction requires many uncertainties to be addressed, such as illumination noise, sudden changes in daylight levels and structural changes. The use of fuzzy logic is suggested in the studies of Fida E. Baf [10, 11, 12] as a means of eliminating such uncertainties, specifically the pixel-based fuzzy background subtraction approach. In these studies, the feature vector at each pixel contains three elements: the values of the pixel in the Cb, Cr color band, and texture information obtained from the Local Binary Pattern which describe in [13]. Fuzzy similarity criteria used for determining the similarities of these three values, between current frame and background frame. The similarity measure can be expressed as a single similarity by combining these three similarity criteria by a discrete Choquet integral.

This study proposes an effective fuzzy background subtraction method, specifically for traffic surveillance applications, and evaluates the method by applying it to videos taken from actual traffic surveillance cameras. The method is defined in the second section, which contains sub-headings of Feature Represent and Similarity, Aggregation Operator and Background Updating. The results of the application are presented in the third section, and conclusions are made in section four.

2 Fuzzy Background Subtraction: A Region Based Method

The key stages in the generic background subtraction framework are: determining the background model, determining how to assign the first value to the background, determining the feature vector to be used, and determining how to update the background [14]. The steps to be applied in this study are presented in Fig1.

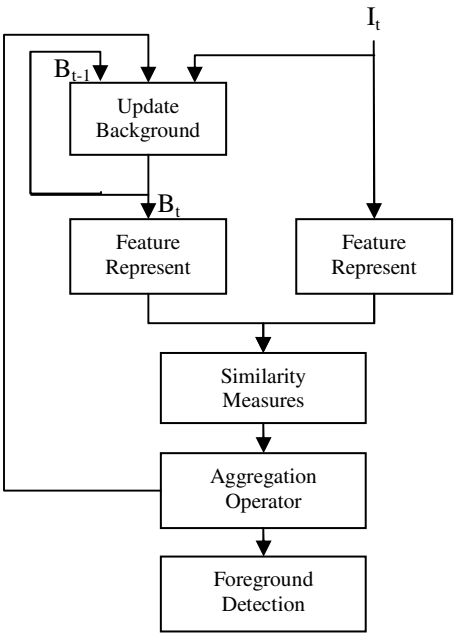


Fig. 1. Diagram of Background Subtraction

Symbols I and B in the diagram represent respectively the current frame taken from the video and background frame, while t denotes the time index.

2.1 Feature Represent and Similarity

Feature extraction is the most important stage in machine learning algorithms [15], and is also the most influential step in background/foreground classification during background subtraction. The features to be determined on this step should allow the above-mentioned separation. In previous literature, color components in RGB, YCbCr and HSV spaces have been used, although some studies have adopted the Ohta color space such as [16]. In addition to color component, features such as texture, edges, corners and motion are also influential [17], and so in this study, aside from magnitude and angle of luminance vector; information on motion is also determined as features.

Luminance Vector. Since information directly taken from a camera sensors are in the grayscale or RGB color space, these color bands are widely used as features, however, they may be affected significantly by changes in lighting levels, in that if grayscale or RGB components are evaluated pixel based, they will not represent the global information about the object. Luminance vector is a 1x25 row vector, which is consisted of 5x5 sub-region's grayscale value, and it is expressed as Eq1.

$$\vec{L}^r = \left\{ I_{x-2,y-2} \quad I_{x-2,y-1} \quad .... \quad I_{x,y} \quad .... \quad I_{x+2,y+1} \quad I_{x+2,y+2} \right\}_{1x25} \tag{1}$$

$\vec{L}^r$, I represent the luminance vector of the sub-region r and luminance of image respectively in Eq1, in which x and y refers to pixel index of image, also (x, y) is the center of sub-region r.

Magnitude Component. Distance from end point of vector to origin point is called magnitude of vector, and it is calculate as Eq.2. Furthermore it referred to as the norm of the vector. In Eq.2 $\vec{L}$ represents the luminance vector, $\| \ \|$ represents norm operator and i refers the index of vector.

$$\left\| \vec{L} \right\| = \sqrt{\sum_i L_i^2} \tag{2}$$

The similarity of two scalar or vectors is a function of the distance between them, and so measurements of similarity can be defined as many as distance measures. Euclid, Manhattan, Cosine and Mahalanobis distances are all commonly used forms of measurement. A ratio-based similarity function is used for scalar feature. According to the study [16], the ratio based similarity is calculated by dividing the smaller value by the greater value. Calculation of the ratio based similarity is shown in Eq3.

$$M(\vec{L}_b, \vec{L}_c) = \min\left(\left\| \vec{L}_b \right\|, \left\| \vec{L}_c \right\| \right) / \max\left(\left\| \vec{L}_b \right\|, \left\| \vec{L}_c \right\| \right) \tag{3}$$

L, M represent the luminance vector and similarity of magnitude function respectively. Addition to this, b and c sub-scripts represent the background and current frames respectively.

Angle Component. Although, magnitude of luminance vector expressed important feature of sub-region, But it is directly affected by the illumination change definitely. On the other hand, angle of luminance vector is define as angles difference between luminance vector to reference vector, and it is not affected by any scale factor (illumination change). The cosine of angles difference between two vectors is calculated as Eq4.

$$A(\vec{L}_b, \vec{L}_c) = \frac{\sum_i L_b^i . L_c^i}{\left\| \vec{L}_b \right\| . \left\| \vec{L}_c \right\|} \tag{4}$$

In Eq.4, A and L represents the cosine of angle function (cosine similarity function), and luminance vector respectively. Index of i , b, c represents index of vector, background and current frames respectively.

In proposed methods, addition to the difference between background and current frame, the difference between current frame and previous frame also used for motion information. For that reason, general similarity vector represents as Eq5.

$$S^r = \left\{ M(\vec{L}_B^r, \vec{L}_{I_t}^r) \quad A(\vec{L}_B^r, \vec{L}_{I_t}^r) \quad M(\vec{L}_{I_{t-1}}^r, \vec{L}_{I_t}^r) \quad A(\vec{L}_{I_{t-1}}^r, \vec{L}_{I_t}^r) \right\} \tag{5}$$

S^r, L^r, M, A represent general similarity vector of region r, luminance vector of region r, magnitude similarity, cosine similarity respectively. B, I_t, I_{t-1} sub-scripts represent background, current frame and previous frame respectively.

2.2 Aggregation Operator

The combining operator is a function that degrades a set of numbers to a single representative, or a meaningful number [18]. The combining operator maps a vector of the size *n*, of which the components are taken from a specific set to an element in this set. Thus, pieces of information obtained from various sources can be used at the same time [19]. For example, in evaluations of weight, width or length of an object, or ratios of how much heavier, wider or longer one object is when compared to another, the arithmetic, geometric or harmonic averages of these specialties can be calculated to obtain a general idea of these features in what can be referred to as a combining process. The most widely known and commonly used combining operator for the combining of numerical information is the arithmetic average calculation, which was first formulated by Aczel in 1984. Such operators calculate the linear combination of a given set of values using a set of weights representing the importance or reliability of the source from where the values were obtained. This method leads to satisfactory results when resources are independent, in other words, when there is no dependency between the resources [18]. In cases where there is a dependency, it may be preferable to use statistical combination operators or fuzzy integrals for the combination process [20].

The approaches of Bayes and Dempster-Schafer are the most widely known methods of resource combination [21]. In statistical combinations, it is necessary for the statistics from all sources to be clear. While all statistics are required for the Bayes approach, Dempster-Schafer approach only needs upper and lower limits of statistics. Clarity of the statistics is not necessary [22].

Sugeno or Choquet integrals can be used when no statistics are available, but when there is some expert knowledge [23]. According to Zhang and Xu, the Choquet integral is more effective when the values of the resources to be combined to have a meaning with each other, and thus these resources are in a cardinal relationship. On the other hand, the Sugeno integral is satisfactory when the arrangement of resources is important, and thus in an ordinal relationship. For the purposes of this study, the Choquet integral is used, and the method is presented in the following section.

Choquet Integral. Choquet and Sugeno integrals are the two basic classes of fuzzy integrals, the weights of which are the calculated fuzzy measures that explain the

dependency between resources [24]. For this reason, the effectiveness of redundant resources is reduced. The most important and complex step in this method is the determination of fuzzy measure, which is the functions that map all power sets of resource to intervals of [0,1]. Fuzzy measure is defined as follows:

Let set S, expressing k similarities as $S=\{S_1,S_2,...,S_k\}$. If the power set representing all sub-sets of set S is shown as P(S), g, the function of the fuzzy measure is expressed as in Eq.6.

$$g : P(S) \rightarrow [0,1] \tag{6}$$

Additionally, function g is required to satisfy the three items specified in Eq.7.

$$\begin{gathered} g(\varnothing) = 0 \\ g(S) = 1 \\ A, B \subseteq S \; and \; A \subseteq B \rightarrow g(A) \leq g(B) \end{gathered} \tag{7}$$

When k is the number of resources, function g is required to satisfy condition for 2^k discrete inputs. Here, it is important that the identification of function g be made by an expert who can express the effect of resources on the results. Once the fuzzy measure function has been determined, the Choquet integral is calculated. The expression of the integral is given below.

By defining the similarity vector with a number of k of the pixel in the specific region to combine similarities, as $S=\{S_1,S_2,...,S_k\}$, let the vector in which the components are arranged in ascending order be as in Eq.8.

$$\hat{S} = \left\{S_{(1)}, S_{(2)},..., S_{(k)}\right\} \qquad S_{(1)} \leq S_{(2)} \leq ... \leq S_{(k)} \tag{8}$$

The Choquet integral value is calculated as in Eq.9, where g represents the fuzzy measure function.

$$C = \sum_{i=1}^{k} S_{(i)}.\left[g\left(\{S_{(i)},..,S_{(k)}\}\right) - g\left(\{S_{(i+1)},..,S_{(k)}\}\right)\right] \tag{9}$$

C is used in the combination of similarity value of the interested region. When index value i, is equal to k in the summation expression, the second value in the square brackets shall be the fuzzy measure value of the set from the element from k+1 to k. The result shall be a null set, since k+1 is greater than k. Accordingly, the fuzzy measure value of the null set shall be 0, as defined in the first statement of Eq.7.

2.3 Background Update

Update methods, refer to how the background updates itself automatically, for which three methods have been defined in previous literature: blind, selective and adaptive [11]. In the blind background update method; new background pixel value is

considered to be a linear combination of the current frame and previous background frame. In this case, whether the related pixel of the current frames is the background or foreground is meaningless, making this approach unsuccessful [25]. Since pixels known to be in the foreground distort the background, selective methods, updated in different linear combinations depending on whether the related pixel belongs to the background or the foreground, are developed. Since the determination of whether the pixel classification is binary condition (background or foreground), uncertainties in the problem are far from eliminated.

The fuzzy background update process is defined in Eq.10. By μF and μB, being membership functions of foreground and background fuzzy sets respectively, with, I and B being current and background frames respectively, being learned coefficient and r being the sub-region of the frame [11].

$$B_t^r = \mu F(C^r).B_{t-1}^r + \mu B(C^r).\left((1-\alpha).B_{t-1}^r + \alpha.I_t^r\right) \tag{10}$$

The $\mu F(x) = 1 - \mu B(x)$ relationship being valid, both functions are a function of the similarity value obtained through the Choquet integral.

3 Results

The suggested method assumes that the background is completely known at the beginning of iterations; however, this may not be possible for all applications. In such cases, the background is first estimated using median value of each pixel through 300 iterations, after proposed method can be applied. The main difference of this study from previous examples in literature is its use throughout the region based feature vector instead of pixels based approach, and its use different feature vector and different similarity measure too.

Table 1. Fuzzy Measure Value

subsets	Fuzzy outputs	subsets	Fuzzy outputs
{0}	0	{S_2 ,S_3}	0.9
{S_1}	0.7	{S_2 ,S_4}	0.9
{S_2}	0.8	{S_3 ,S_4}	0.95
{S_3}	0.7	{S_1,,S_2,S_3}	0.93
{S_4}	0.8	{S_1,,S_2,S_4}	0.94
{S_1,,S_2}	0.9	{S_1,,S_3,S_4}	0.96
{S_1 ,S_3}	0.9	{S_2,,S_3,S_4}	0.98
{S_1 ,S_4}	0.9	{S_1,,S_2,,S_3,S_4}	1

In addition, the 16-element fuzzy measure function suggested for the combining of four different features is also unique in this study, and it is shown in Table1.

Various performance criteria have been suggested in literature to evaluate how well the background has been determined [11]. The expression of metric used in this evaluation is given in Eq.11 [26].

$$P(B,\widehat{B}) = \frac{B \cap \widehat{B}}{B \cup \widehat{B}} \tag{11}$$

Here, $B, \widehat{B}$ denote respectively the ground-truth value of the background and the background predicted by the method, while P denotes performance. As can be seen in Eq.11, the success criteria is in the interval of [0 1]. The performances of our method and Gauss Mixture Model (GMM) which explain in [5], are given in Table 2, and the representative outputs are shown in Fig.2.

Fig. 2. a) Sample frame of video, b) Background with proposed method, c) Foreground objects

Table 2. Performance of Methods

	GMM Method	Fuzzy Pixel Based Method	Fuzzy Region Based Method
Performance	%82	%85	%93
Time Consumption	0.015sn/fr	0.019sn/fr	0.1sn/fr

Tests were performed on PC, which have four cores 2.20 GHz CPU, and 6GB Ram. The length of the video is nearly 20 minutes. Resolution of image is 576x720 and color space is RBG. All the videos obtained from [27]. According to Table.1, region based fuzzy background subtraction methods performance is the highest. On the other hand, because of the region process, the method needs much more CPU time.

4 Conclusion

Traffic cameras are today in common use for the recording of statistics at intersections, junctions and highways; however, extraction of statistical parameters which are necessary for traffic simulation models, from these cameras is both time-consuming

and unreliable. Processing camera data automatically and identifying the required parameters is not always possible due to such factors as environmental noise, excessive traffic intensity in the case of traffic jams and sudden changes in light intensity. The method suggested throughout this study constitutes a new approach to getting over these difficulties, the success of which has been proven in empirical studies. Rule-based methods and their effects on background subtraction by a type-2 fuzzy sets are to be examined in further studies.

Acknowledgement. This work is supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under the project 'Hybrid models of neural network method for road safety regulations: Safety index calibration and Intelligent Transportation Systems based safety control' with no. 108M299.

References

1. Bouwmans, T., El Baf, F., Vachon, B.: Background Modeling using Mixture of Gaussians for Foreground Detection - A Survey. Recent Patents on Computer Science 1(3), 219–237 (2008)
2. Lo, B.P.L., Velastin, S.A.: Automatic congestion detection system for underground platforms. In: Intelligent Multimedia, Video and Speech Processing, pp. 158–161. IEEE Press, Hong Kong (2001)
3. Cucchiara, R., Grana, C., Piccardi, M., Prati, A.: Detecting moving objects, ghosts, and shadows in video streams. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(10), 1337–1342 (2003)
4. Wren, C., Azarbayejani, A., Darrell, T., Pentland, A.: Pfinder: real-time tracking of the human body. In: International Conference on Automatic Face and Gesture Recognition, pp. 51–56. IEEE Press (1996)
5. Stauffer, C., Grimson, W.E.L.: Adaptive background mixture models for real-time tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 246–252. IEEE Press (1999)
6. Han, B., Comaniciu, D., Davis, L.S.: Sequential Kernel Density Approximation through Mode Propagation: Applications to Background Modeling. In: Proc. Asian Conf. Computer Vision (2004)
7. Karmann, K., Brandt, A., Gerl, R.: Moving object segmentation based on adaptive reference images. Signal Process. Theor. Appl. 1(5), 951–954 (1990)
8. Balcilar, M., Sonmez, A.C.: Extracting Vehicle Density From Background Estimation Using Kalman Filter. In: 23rd International Symposium on Computer and Information Sciences, pp. 504–508. IEEE Press, Istanbul (2008)
9. Jang, J.S.R., Sun, C.T., Mizutani, E.: Neuro-Fuzzy and Soft Somputing, New Jersey (1997)
10. El Baf, F., Bouwmans, T., Vachon, B.: Foreground Detection Using the Choquet Integral. In: Image Analysis for Multimedia Interactive Services, pp. 187–190. IEEE Press (2008)
11. El Baf, F., Bouwmans, T., Vachon, B.: A fuzzy approach for background subtraction. In: International Conference on Image Processing, pp. 2648–2651. IEEE Press (2008)
12. El Baf, F., Bouwmans, T., Vachon, B.: Fuzzy foreground detection for infrared videos. In: Computer Vision and Pattern Recognition Workshops, pp. 1–6. IEEE Press (2008)

13. Heikkila, M., Pietikinen, M.: A texture-based method for modeling the background and detecting moving objects. IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI 28(4), 657–662 (2006)
14. El Baf, F., Bouwmans, T., Vachon, B.: Fuzzy integral for moving object detection. In: Fuzzy Systems (IEEE World Congress on Computational Intelligence), pp. 1729–1736. IEEE Press (2008)
15. Alpaydin, E.: Introduction to Machine Learning. MIT Press (2004)
16. Zhang, H., Xu, D.: Fusing Color and Texture Features for Background Model. In: Wang, L., Jiao, L., Shi, G., Li, X., Liu, J. (eds.) FSKD 2006. LNCS (LNAI), vol. 4223, pp. 887–893. Springer, Heidelberg (2006)
17. Gonzalez, R.C., Woods, R.E., Eddins, S.L.: Digital Image Processing Using Matlab. Prentice Hall, Upper Saddle River (2004)
18. Gunay, H.: Aggreagation operators in fuzzy decision making and applications, Master Thesis, Yildiz Technical University (2006) (in Turkish)
19. Detyniecki, M.: Mathematical aggregation operators and their application to video querying. Doctoral thesis, Laboratoire d'Informatique de Paris (2000)
20. Godo, L., Torra, V.: Extending Choquet Integrals for Aggregation of Ordinal Values. In: Proc. of the Eighth Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 410–417, Madrid (2000)
21. Zhao, W., Fang, T., Jiang, Y.: Data Fusion Using Improved Dempster-Shafer Evidence Theory for Vehicle Detection. In: Fourth International Conference on Fuzzy Systems and Knowledge Discovery, pp. 487–491 (2007)
22. Koks, D.: An Introduction to Bayesian and Dempster-Shafer Data Fusion. Engineering 175(2), 1–52 (2003)
23. Grabisch, M., Roubens, M.: Application of the Choquet integral in multicriteria decision making. In: Grabisch, M., Murofushi, T., Sugeno, M. (eds.) Fuzzy Measures and Integrals: Theory and Applications. STUDFUZZ, pp. 348–374. Physica Verlag, New York (2000)
24. Marichal, J.L.: Tolerant or intolerant character of interacting criteria in aggregation by the Choquet integral. European Journal of Operational Research 155(3), 771–791 (2004)
25. Balcilar, M.: Extracting traffic flow velocity and vehicle density within MPEG videos using motion vectors and background estimation technique, Master Thesis, Yildiz Technical University (2007) (in Turkish)
26. Li, L., Huang, W.: Statistical Modeling of Complex Background for Foreground Object Detection. IEEE Transaction Image Processing 13(11), 1459–1472 (2004)
27. Çelikoğlu, H.B., Cığızoğlu, H.K., Gürcanlı, G.E.: Hybrid models of neural network method for road safety regulations: Safety index calibration and Intelligent Transportation Systems based safety control. Final report of the project with no. 108M299 supported by the Scientific and Technological Research Council of Turkey (TUBITAK). TUBITAK, Ankara (2010) (in Turkish)

A Robust Fuzzy Adaptive Control Algorithm for a Class of Nonlinear Systems

Sašo Blažič and Igor Škrjanc

University of Ljubljana, Faculty of Electrical Engineering
Tržaška 25, 1000 Ljubljana, Slovenia
{saso.blazic,igor.skrjanc}@fe.uni-lj.si

Abstract. The paper presents a general methodology of adaptive control based on soft computing models to deal with unknown plants. The problem of parameter estimation is solved using a direct approach, i.e., the controller parameters are adapted without explicitly estimating plant parameters. Thus, very simple adaptive and control laws are constructed within the Lyapunov stability framework. The proposed control ensures global stability of the overall system and convergence of the tracking error to a residual set that depends on the size of unmodelled dynamics. The generality of the approach is substantiated by Stone-Weierstrass theorem, which indicates that any continuous function can be approximated by fuzzy basis function expansion. The hallmarks of the approach are its simplicity and transparency. The paper shows the efficiency of the proposed approach on the control of a heat exchanger.

Keywords: adaptive control, fuzzy model, Takagi-Sugeno model, model-reference adaptive control.

1 Introduction

Many successful applications of fuzzy and neural network-based controllers [9,11,8] have shown their ability to control nonlinear plants. A lot of effort has been put to neuro-fuzzy identification of complex plants, which cannot be easily theoretically modelled. Based on neuro-fuzzy representation of the plant dynamics the neuro-fuzzy adaptive control approaches appeared in literature [6] where a detailed discussion of identification and control of dynamical systems based on neural networks is given. In [15] and [12] a stable adaptive fuzzy controller for nonlinear systems is designed and explained; in [7] an adaptive control using multiple models is developed and investigated. A direct adaptive fuzzy-model-based control algorithm is presented in [1]. In this case the controller is based on an inverse semi-linguistic fuzzy process model which is identified and adapted via input-matching technique using a general gradient-descent algorithm.

In this paper, robustness analysis of the controlled system plays a central role. The combination of adaptive control theory based on models obtained by fuzzy basis function expansion results in direct model-reference fuzzy adaptive control which provides higher adaptation ability than basic adaptive control systems.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 297–306, 2013.

The proposed control algorithm is an extension of direct model-reference fuzzy adaptive control to nonlinear plants. Direct fuzzy adaptive control directly adjusts the parameters of a fuzzy controller to achieve approximate asymptotic tracking of the model-reference input. The generality of the proposed algorithm is substantiated by Stone-Weierstrass theorem which indicates that any continuous function can be approximated by fuzzy basis function expansion. This reconstruction error acts as a disturbance in the adaptive law. Robust adaptive control was proposed to overcome the problem of disturbances and unmodelled dynamics [4]. Similar solutions have also been used in adaptive fuzzy and neural controllers, i.e. projection [14], dead zone [5], leakage [3], adaptive fuzzy backstepping control [13] etc. have been included in the adaptive law to prevent instability due to reconstruction error. In this paper, not only reconstruction error and disturbances but also error due to high-order parasitic dynamics (which are inevitable) is treated explicitly. The latter is especially problematic since it can become unbounded [4]. The proposed control ensures global stability of the overall system and convergence of the tracking error to the residual set that depends on the unmodelled dynamics. The rationale behind the study of the influence of parasitics is that the control plant is assumed to be nonlinear and predominantly of the first order (higher-order parasitics are catered for by the robustness properties of the controller). In our opinion, such plants occur quite often in process industries.

In section 2 the class of plants under investigation is presented, in Section 3 the control algorithm is given, in Section 4 the stability issues are discussed, in Section 5 the plant model is depicted, and in Section 6 the results are presented.

2 The Class of Nonlinear Plants

Our goal is to design control for a class of plants that include nonlinear time-invariant systems where the model behaves similarly to a first-order system at low frequencies. If the plant were the first-order system, it could be described by a fuzzy model in the form of if-then rules:

$$\begin{aligned} &\textbf{if } z_1 \textbf{ is } A_{i_a} \textbf{ and } z_2 \textbf{ is } B_{i_b} \textbf{ then } \dot{y}_p = -a_i y_p + b_i u \\ &i_a = 1, \ldots, n_a \quad i_b = 1, \ldots, n_b \quad i = 1, \ldots, k \end{aligned} \tag{1}$$

where u and y_p are the input and the output of the plant respectively, A_{i_a} and B_{i_b} are fuzzy membership functions, and a_i and b_i are the plant parameters in the i-th fuzzy domain. The antecedent variables that define the fuzzy domain in which the system is currently situated are denoted by z_1 and z_2 (actually there can be only one such variable or there can also be more of them, but this does not affect the approach described in this paper). There are n_a and n_b membership functions for the first and the second antecedent variables, respectively. The product $k = n_a \times n_b$ defines the number of fuzzy rules. The membership functions have to cover the whole operating area of the system. The output of the Takagi-Sugeno model is then given by the following equation

$$\dot{y}_p = \frac{\sum_{i=1}^{k} \left[\beta_i^0(\boldsymbol{\varphi})(-a_i y_p + b_i u)\right]}{\sum_{i=1}^{k} \beta_i^0(\boldsymbol{\varphi})} \tag{2}$$

where $\boldsymbol{\varphi}$ represents the vector of antecedent variables z_i (in the case of fuzzy model given by Eq. (1), $\boldsymbol{\varphi} = [z_1 \;\; z_2]^T$). The degree of fulfilment $\beta_i^0(\boldsymbol{\varphi})$ is obtained using the T-norm, which in this case is a simple algebraic product of membership functions

$$\beta_i^0(\boldsymbol{\varphi}) = T(\mu_{A_{i_a}}(z_1), \mu_{B_{i_b}}(z_2)) = \mu_{A_{i_a}}(z_1) \cdot \mu_{B_{i_b}}(z_2) \tag{3}$$

where $\mu_{A_{i_a}}(z_1)$ and $\mu_{B_{i_b}}(z_2)$ stand for degrees of fulfilment of the corresponding fuzzy rule. The degrees of fulfilment for the whole set of fuzzy rules can be written in a compact normalised form as

$$\boldsymbol{\beta}^T = \frac{\left[\beta_1^0 \; \beta_2^0 \; \ldots \; \beta_k^0\right]^T}{\sum_{i=1}^{k} \beta_i^0} \in \mathbb{R}^k \tag{4}$$

Due to (2) and (4), the first-order plant can be modelled in fuzzy form as

$$\dot{y}_p = -(\boldsymbol{\beta}^T \mathbf{a}) y_p + (\boldsymbol{\beta}^T \mathbf{b}) u \tag{5}$$

where $\mathbf{a} = \left[a_1 \; a_2 \; \ldots \; a_k\right]^T$ and $\mathbf{b} = \left[b_1 \; b_2 \; \cdots \; b_k\right]^T$ are vectors of unknown plant parameters in respective fuzzy domains ($\mathbf{a}, \mathbf{b} \in \mathbb{R}^k$).

To assume that the controlled system is of the first order is a quite huge idealisation. Parasitic dynamics and disturbances are therefore included in the model of the plant. The fuzzy model of the first order is generalised by adding stable factor plant perturbations and disturbances, which results in a following model [2]:

$$\dot{y}_p(t) = -(\boldsymbol{\beta}^T(t)\mathbf{a}) y_p(t) + (\boldsymbol{\beta}^T(t)\mathbf{b}) u(t) - \Delta_y(p) y_p(t) + \Delta_u(p) u(t) + d(t) \tag{6}$$

where p is a differential operator d/dt, $\Delta_y(p)$ and $\Delta_u(p)$ are stable strictly proper linear operators, while d is bounded signal due to disturbances [2].

Eq. (6) represents the class of plants to be controlled by the approach proposed in the following sections. The control is designed based on the model given by Eq. (5) while the robustness properties of the algorithm prevent the instability due to parasitic dynamics and disturbances.

3 The Proposed Fuzzy Adaptive Control Algorithm

The fuzzy model reference adaptive control is proposed in the paper to achieve tracking control for the class of plants described in the previous section. The control goal is that the plant output follows the output y_m of the reference model. The latter is defined by a first order linear system $G_m(p)$:

$$y_m(t) = G_m(p) w(t) = \frac{b_m}{p + a_m} w(t) \tag{7}$$

where $w(t)$ is the reference signal while b_m and a_m are the constants that define desired behaviour of the closed system. The tracking error

$$\varepsilon(t) = y_p(t) - y_m(t) \tag{8}$$

therefore represents some measure of the control quality. To solve the control problem simple control and adaptive laws are proposed in the following subsections.

3.1 Control Law

The control law is the same as the one proposed in [2]:

$$u(t) = \left(\boldsymbol{\beta}^T(t)\hat{\mathbf{f}}(t)\right) w(t) - \left(\boldsymbol{\beta}^T(t)\hat{\mathbf{q}}(t)\right) y_p(t) \tag{9}$$

where $\hat{\mathbf{f}}(t) \in \mathbb{R}^k$ and $\hat{\mathbf{q}}(t) \in \mathbb{R}^k$ are the control gain vectors to be determined by the adaptive law. This control law is obtained by generalising the model reference adaptive control algorithm for the first order linear plant to the fuzzy case.

3.2 Adaptive Law

The adaptive law proposed in this paper is based on the adaptive law from [2]. The e_1-modification was used in the leakage term in [2]. The alternative approach is proposed here:

$$\begin{aligned} \dot{\hat{f}}_i &= -\gamma_{fi} b_{sign} \varepsilon w \beta_i - \gamma_{fi} \sigma' w^2 \beta_i^2 (\hat{f}_i - f_i^*) \quad i = 1, 2, \ldots k \\ \dot{\hat{q}}_i &= \gamma_{qi} b_{sign} \varepsilon y_p \beta_i - \gamma_{qi} \sigma' y_p^2 \beta_i^2 (\hat{q}_i - q_i^*) \quad i = 1, 2, \ldots k \end{aligned} \tag{10}$$

where γ_{fi} and γ_{qi} are positive scalars referred to as adaptive gains, $\sigma' > 0$ is the parameter of the leakage term, f_i^* and q_i^* are the a priori estimates of the control gains $\hat{f}_i$ and $\hat{q}_i$, respectively, and b_{sign} is the sign of all the elements in vector $\mathbf{b}$. If the signs of all elements in vector $\mathbf{b}$ are not the same, the plant is not controllable for some $\boldsymbol{\beta}$ ($\boldsymbol{\beta}^T\mathbf{b}$ is equal to 0 for this $\boldsymbol{\beta}$) and the control is not possible using this approach.

It is possible to rewrite the adaptive law (10) in the compact form if the control gain vectors $\hat{\mathbf{f}}$ and $\hat{\mathbf{q}}$ are defined. Then the adaptive law (10) takes the following form:

$$\begin{aligned} \dot{\hat{\mathbf{f}}} &= -\boldsymbol{\Gamma}_f b_{sign} \varepsilon w \boldsymbol{\beta} - \boldsymbol{\Gamma}_f \sigma' w^2 \operatorname{diag}(\boldsymbol{\beta}) \operatorname{diag}(\boldsymbol{\beta})(\hat{\mathbf{f}} - \hat{\mathbf{f}}^*) \\ \dot{\hat{\mathbf{q}}} &= \boldsymbol{\Gamma}_q b_{sign} \varepsilon y_p \boldsymbol{\beta} - \boldsymbol{\Gamma}_q \sigma' y_p^2 \operatorname{diag}(\boldsymbol{\beta}) \operatorname{diag}(\boldsymbol{\beta})(\hat{\mathbf{q}} - \hat{\mathbf{q}}^*) \end{aligned} \tag{11}$$

where $\boldsymbol{\Gamma}_f \in \mathbb{R}^k \times \mathbb{R}^k$ and $\boldsymbol{\Gamma}_q \in \mathbb{R}^k \times \mathbb{R}^k$ are positive definite matrices, $\operatorname{diag}(\mathbf{x}) \in \mathbb{R}^k \times \mathbb{R}^k$ is the diagonal matrix with the elements of vector $\mathbf{x}$ on the main diagonal, while $\hat{\mathbf{f}}^* \in \mathbb{R}^k$ and $\hat{\mathbf{q}}^* \in \mathbb{R}^k$ are the a priori estimates of the control gain vectors.

4 Stability Issues of the Proposed Control Algorithm

It is not possible in general to find vectors $\hat{\mathbf{f}}$ and $\hat{\mathbf{q}}$ that would permit zero tracking error in each operating point since fuzzy modelling only guarantees arbitrary small tracking errors. This means that we should always study the influence of modelling errors. In the case of unmodelled dynamics the adaptive schemes may easily go unstable. The lack of robustness is primarily due to the adaptive law which is nonlinear in general and therefore more susceptible to modelling error effect.

The robustness of the adaptive control scheme can be improved by converting the pure integral action of the adaptive law to a leaky integration (this extension is referred to as the leakage modification). But an arbitrary leakage (e.g., sigma-modification) alone is not enough to cope with the dynamical unmodelled dynamics such as parasitics that can result in an unbounded modelling error. Dynamical signal normalisation is utilized for this reason. In our approach, the normalisation is introduced through a quadratic term in the leakage which results in the globally stable system.

It can be proven that the proposed approach results in the following properties of the controlled system:

$$\begin{gathered} \epsilon, \hat{\mathbf{f}}, \hat{\mathbf{q}}, \dot{\hat{\mathbf{f}}}, \dot{\hat{\mathbf{q}}} \in \mathcal{L}_\infty \\ \epsilon \in \mathcal{S}(\sigma'^2 + ||\Delta_u(s)||_2^2 + ||\Delta_y(s)||_2^2 + \bar{d}^2) \end{gathered} \tag{12}$$

if the reference signal is continuous and the antecedent variables are continuous. Comparing this result with the algorithm in [2], there are less restrictions on the part of the complex plane where $\Delta_u(s)$ and $\Delta_y(s)$ should be analytic. This is due to a new form of the adaptive law.

5 The Model of a Nonlinear Heat-Exchanger Plant

The simulation study was done on a heat-exchanger plant control which together with sensors and actuators limitation represents a serious problem from the point of optimal energy consumption. The problem lies in the nonlinearity of the system. The object of our investigation, a real temperature plant, consists of: a plate heat-exchanger, a reservoir with heated water, two thermocouples and a motor driven valve. The plate heat exchanger, through which hot water from an electrically heated reservoir is continuously circulating in the countercurrent flow to cold process fluid (cold water). The thermocouples are located at the inlet and outlet flows of the exchanger. Power to the heater may be controlled by time proportioning control using the external control loop. The flow of the heating fluid can be controlled by the proportional motor driven valve. A schematic diagram of the plant is shown in Fig. 1. The temperature of heated water $T_{sp}(k)$ is measured on the temperature sensor TC4 which is at the outlet of the secondary circuit, the temperature of cold water at the inlet of secondary circuit $T_{ep}(k)$ is measured on the temperature sensor TC3 and $T_{ec}(k)$

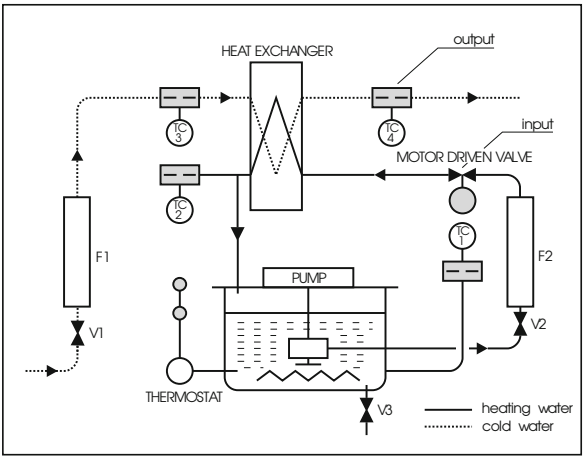


Fig. 1. The heat-exchanger pilot plant

represents the temperature of hot water at the inlet of the primary circuit which is measured on the temperature sensor TC1. The primary circuit flow $F_c(k)$ is measured on optical flow sensor F2 and is defined by motor driven valve and the secondary flow $F_p(k)$ is measured on the optical flow sensor F1.

The controlled variable of our problem is the temperature in the secondary circuit T_{sp} which is manipulated with the flow F_c. The heat-exchanger is just one part of the plant, so the sensors and the actuators should also be modelled. For nonlinear systems with well-understood physical phenomena fundamental modelling is preferable. Although the physical phenomena in the case of heat-exchanger are well investigated, there are still some physical parameters which should be estimated assuming a certain structure of the process dynamics. The simplified first-principle model of heat-exchanger is described by the following differential equations

$$\tau_2(T_{sp})\dot{T}_{sp} + T_{sp} = \gamma T_{ep} + (1-\gamma)T_{ec} \tag{13}$$

where the generalized formula for γ is given in the literature [10] and can be written as

$$\gamma = \frac{1 + k_c(\frac{1}{F_c})^m}{1 + k_c((\frac{1}{F_c})^m + (\frac{1}{F_p})^m)} \tag{14}$$

where k_c and m are unknown constants and τ_2 is an unknown function of operating point. All those parameters are unknown and should be estimated somehow.

Our main goal is to control the temperature T_{sp} by changing the primary circuit flow F_p. Although the process is very complex, it could be presented as a model with approximately first order dynamics. It should be noted, however, that parasitic dynamics are also present as a consequence of actuators, sensors, heat junctions, mass flows etc.

6 Simulation Study of Fuzzy Model-Reference Adaptive Control

Our goal was to design control algorithm that would enable that the closed-loop system behaves as close to a linear reference model as possible. Two different approaches were compared: fuzzy model reference adaptive control (FMRAC) described by Eqs. (9) and (11), and classical model reference adaptive control for linear plants. To make the latter robust to unmodelled dynamics, disturbances, and noise, the robust adaptive law with e_1-modification was used.

The simulation study will be described next. The reference signal was periodic and piece-wise constant. As proposed in the paper, the first order reference model was chosen. Even if the plant has high order parasitics, it was forced to follow reference model, described by transfer function

$$G_m(s) = \frac{0.01}{s + 0.01} \tag{15}$$

It is known a priori that perfect following cannot be achieved, but the results will show that this simplification is justified.

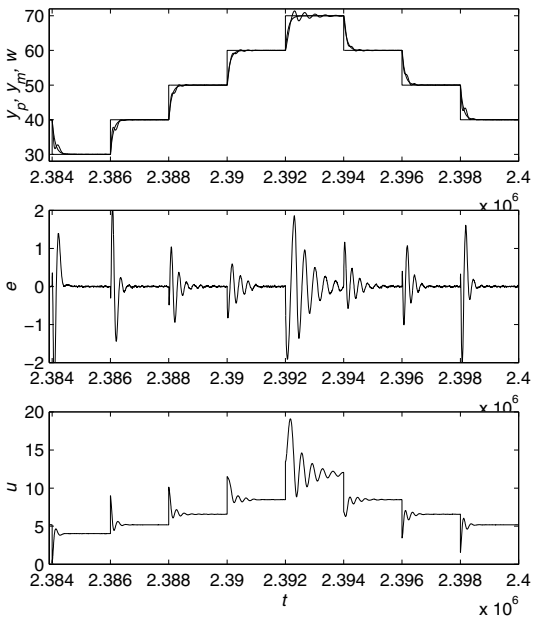


Fig. 2. The last period of the signals in the MRAC case

The choice of proper adaptive gain is very important when one deals with adaptive systems. In nonlinear case this is even more obvious, since adaptive parameters do not converge to specific values. Rather, they tend to change when

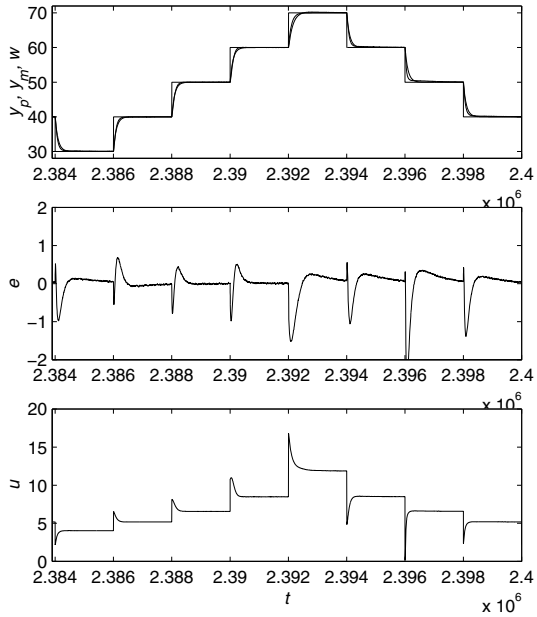


Fig. 3. The last period of the signals in the FMRAC case

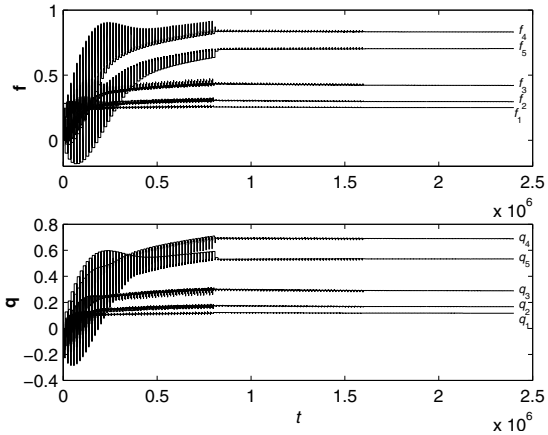


Fig. 4. The trajectories of the fuzzified adaptive parameters

the operating point changes. Our approach reduces these oscillations since different parameters are estimated in different fuzzy domains. Unfortunately, fuzzy model fails to achieve perfect modelling of the plant, and the parameters still oscillate a little. These oscillations can be even further amplified by choosing adaptive gains too high. If, on the other hand, the adaptive gains are too low,

the estimation is too slow, and, especially in the linear MRAC case, the parameters fail to follow rapid changes of the operating point. This problem is not so serious in the fuzzy adaptive case, since fuzzy domains have to be chosen such that the plant parameters do not differ much in the neighbouring domain. The following adaptive gains were used in our experiments: $\gamma_f = 10^{-5}$ and $\gamma_q = 10^{-5}$. The choice of the leakage parameter was not so critical in this case and $\sigma' = 1$ was used. The adaptive parameters were initialized to 0 in all cases.

First, classical MRAC was tested. It turned out that the adaptive gain was too high in the beginning and the response of the system was not acceptable. After reducing adaptive gains, the parameters converged slower, but quite good performance was achieved. Reducing adaptive gains further made the results worse. The optimal values are used in the experiments. Figure 2 shows the signals after a longer period. In the upper part of the figure, controlled variable is shown, together with reference signal and reference model output. To see the difference more clear, tracking error is shown in the middle part of the figure. The manipulated variable is shown in the lower part of the figure. It can be seen that the results are acceptable, but the system start to oscillate when reference signal becomes large. These oscillations are also present in other operating points, they can be seen easier from the manipulated variable.

The proposed approach was also tested under the same conditions. The original adaptive gain was used in the beginning of the experiment. The estimated parameters were quite oscillatory, but that did not affect plant responses. The plant response at the end of the experiment is shown in Fig. 3. By comparing the signals in Fig. 2 and Fig. 3, it can be said that the latter are more acceptable, since almost perfect tracking is obtained, and the unwanted oscillations in the manipulated variable are not present. One has to realize that the actuator in this case is a pump, and oscillations on the manipulated variable are not admissible. Note the tracking error when the reference signal changes (Fig. 3). It cannot be suppressed by any control algorithm since it is a consequence of the fact that the plant of the second order is forced to follow reference model of the first order.

The adaptive parameters are shown in Fig. 4. Huge oscillations can be seen in the beginning, but the parameters still quasy-settle. Then, the adaptive gains were reduced to obtain better convergence of the parameters. This was done twice in the experiment, and these two points can be identified as a decrease of parameter oscillations in Fig. 4.

7 Conclusion

In this paper a fuzzy model-reference adaptive control system is presented. A novel adaptive law is introduced. It is based on Lyapunov stability analysis. The adaptive parameters of the system are fuzzified. The combination of adaptive control theory based on models obtained by fuzzy basis function expansion results in fuzzy model-reference adaptive control which provides higher adaptation ability than basic adaptive control systems. The proposed leakage term introduces nonlinearity which provides the ability of dynamic signal normalisation.

The main advantage of the proposed approach is simplicity together with high performance. The development of the novel algorithm has been tested using simulation on different nonlinear systems that include unmodelled and unmeasured dynamics.

References

1. Abonyi, J., Andersen, H., Nagy, L., Szeifert, F.: Inverse fuzzy-process-model based direct adaptive control. Mathematics and Computers in Simulation 51(1-2), 119–132 (1999)
2. Blažič, S., Škrjanc, I., Matko, D.: Globally stable direct fuzzy model reference adaptive control. Fuzzy Sets and Systems 139(1), 3–33 (2003)
3. Ge, S., Wang, J.: Robust adaptive neural control for a class of perturbed strict feedback nonlinear systems. IEEE Transactions on Neural Networks 13(6), 1409–1419 (2002)
4. Ioannou, P.A., Sun, J.: Robust Adaptive Control. Prentice-Hall (1996)
5. Koo, K.M.: Stable adaptive fuzzy controller with time varying dead-zone. Fuzzy Sets and Systems 121 (2001)
6. Narendra, K.S., Parthasarathy, K.: Identification and control of dynamical systems using neural networks. IEEE Transactions on Neural Networks 1(1), 4–27 (1990)
7. Narendra, K., Balakrishnan, J.: Adaptive control using multiple models. IEEE Transactions on Automatic Control 42(2), 171–187 (1997)
8. Precup, R.E., Doboli, S., Preitl, S.: Stability analysis and development of a class of fuzzy control systems. Engineering Applications of Artificial Intelligence 13(3), 237–247 (2000)
9. Precup, R.E., Hellendoorn, H.: A survey on industrial applications of fuzzy control. Computers in Industry 62(3), 213–226 (2011)
10. Richalet, J.: Industrial applications of model based predictive control. Automatica 29(5), 1251–1274 (1993)
11. Škrjanc, I., Matko, D.: Predictive functional control based on fuzzy model for heat-exchanger pilot plant. IEEE Transactions on Fuzzy Systems 8(6), 705–712 (2000)
12. Spooner, J., Passino, K.: Stable adaptive control using fuzzy systems and neural networks. IEEE Transactions on Fuzzy Systems 4(3), 339–359 (1996)
13. Tong, S., Li, Y.: Adaptive Fuzzy Output Feedback Tracking Backstepping Control of Strict-Feedback Nonlinear Systems With Unknown Dead Zones. IEEE Transactions on Fuzzy Systems 20(1), 168–180 (2012)
14. Tong, S., Wang, T., Tang, J.T.: Fuzzy adaptive output tracking control of nonlinear systems. Fuzzy Sets and Systems 111 (2000)
15. Wang, L.X.: Stable adaptive fuzzy control of nonlinear systems. IEEE Transactions on Fuzzy Systems 1 (1993)

Disturbance Measurement Utilization in the Efficient MPC Algorithm with Fuzzy Approximations of Nonlinear Models

Piotr M. Marusak

Institute of Control and Computation Engineering, Warsaw University of Technology, ul. Nowowiejska 15/19, 00–665 Warszawa, Poland
P.Marusak@ia.pw.edu.pl

Abstract. Extension of the efficient Model Predictive Control (MPC) algorithm, which uses fuzzy approximations of nonlinear models, with mechanisms of disturbance measurement utilization is proposed. Two methods of disturbance measurement utilization are considered. The first method utilizes a fuzzy model of disturbance influence on the control plant, whereas the second one – a nonlinear model used to obtain the free response. In both methods only the free response generated during the prediction calculation is influenced. Therefore, the prediction has such a form that the MPC algorithm remains to be numerically efficient. Only a quadratic optimization problem must be solved at each iteration in order to derive the control signal. The proposed methods of disturbance measurement utilization can significantly improve control performance offered by the algorithm what is demonstrated in the example control system of a nonlinear chemical CSTR reactor with inverse response.

Keywords: fuzzy control, fuzzy systems, model predictive control, nonlinear control, constrained control.

1 Introduction

The paper describes continuation of research of the algorithm based on fuzzy and nonlinear models proposed in [9]. The algorithm is numerically efficient because it uses the nonlinear model to derive the free response of the control plant (response to past values of the control signal) and the approximate, easy to obtain, fuzzy model to calculate the influence of future control action. Thanks to such an approach the algorithm offers control performance very close to that offered by the algorithm with nonlinear optimization. At the same time, the algorithm is formulated as the easy to solve quadratic optimization problem [9].

In the paper, the efficient fuzzy MPC (FMPC) algorithm is supplemented with the mechanisms of taking the disturbance measurement into consideration. Two methods of disturbance measurement utilization are considered. The first method is designed to be used with an existing (already designed) algorithm, to extend its capabilities; it is based on a fuzzy model of disturbance influence on the process. The second method can be used if the nonlinear model employed

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 307–316, 2013.

to calculate the free response contains both: control and disturbance inputs to the process. Thanks to utilization of the disturbance measurement the control quality offered by the MPC algorithm can be significantly improved.

The next section contains description of the MPC algorithm which utilizes nonlinear and fuzzy models to derive the prediction. In Sect. 3 methods of extension of the algorithm based on fuzzy and nonlinear models are detailed. Results of simulation experiments illustrating efficacy of the proposed approach are presented in Sect. 4. The paper is summarized in the last section.

2 MPC Algorithm Based on Fuzzy and Nonlinear Models

The Model Predictive Control (MPC) algorithms derive future values of manipulated variables predicting future behavior of the control plant many sampling instants ahead. The values of manipulated variables are calculated in such a way that the prediction fulfills assumed criteria. Usually, the minimization of a performance function is demanded subject to the constraints put on manipulated and output variables [3,7,11,13]:

$$\arg\min_{\Delta \boldsymbol{u}} \left\{ J_{\mathrm{MPC}} = \sum_{j=1}^{n_y} \sum_{i=1}^{p} \kappa_j \left(\overline{y}_k^j - y_{k+i|k}^j \right)^2 + \sum_{m=1}^{n_u} \sum_{i=0}^{s-1} \lambda_m \left(\Delta u_{k+i|k}^m \right)^2 \right\} \quad (1)$$

subject to:

$$\Delta \boldsymbol{u}_{\min} \leq \Delta \boldsymbol{u} \leq \Delta \boldsymbol{u}_{\max} \ , \quad (2)$$

$$\boldsymbol{u}_{\min} \leq \boldsymbol{u} \leq \boldsymbol{u}_{\max} \ , \quad (3)$$

$$\boldsymbol{y}_{\min} \leq \boldsymbol{y} \leq \boldsymbol{y}_{\max} \ , \quad (4)$$

where $\overline{y}_k^j$ is a set–point value for the j^{th} output, $y_{k+i|k}^j$ is a value of the j^{th} output for the $(k+i)^{\mathrm{th}}$ sampling instant predicted at the k^{th} sampling instant using a control plant model, $\Delta u_{k+i|k}^m$ are future changes in manipulated variables, $\kappa_j \geq 0$ and $\lambda_m \geq 0$ are weighting coefficients for the predicted control errors of the j^{th} output and for the changes of the m^{th} manipulated variable, respectively; p and s denote prediction and control horizons, respectively; n_y, n_u denote numbers of output and manipulated variables, respectively;

$$\boldsymbol{y} = \left[\boldsymbol{y}^1, \ldots, \boldsymbol{y}^{n_y} \right]^T, \ \boldsymbol{y}^j = \left[y_{k+1|k}^j, \ldots, y_{k+p|k}^j \right] \ , \quad (5)$$

$$\Delta \boldsymbol{u} = \left[\Delta \boldsymbol{u}^1, \ldots, \Delta \boldsymbol{u}^{n_u} \right]^T, \ \Delta \boldsymbol{u}^m = \left[\Delta u_{k|k}^m, \ldots, \Delta u_{k+s-1|k}^m \right] \ , \quad (6)$$

$$\boldsymbol{u} = \left[\boldsymbol{u}^1, \ldots, \boldsymbol{u}^{n_u} \right]^T, \ \boldsymbol{u}^m = \left[u_{k|k}^m, \ldots, u_{k+s-1|k}^m \right] \ , \quad (7)$$

$\Delta \boldsymbol{u}_{\min}$, $\Delta \boldsymbol{u}_{\max}$, $\boldsymbol{u}_{\min}$, $\boldsymbol{u}_{\max}$, $\boldsymbol{y}_{\min}$, $\boldsymbol{y}_{\max}$ are vectors of lower and upper bounds of changes and values of the control signals and of the values of output variables, respectively. As a solution to the optimization problem (1)–(4) the optimal vector of changes in the manipulated variables $\Delta \boldsymbol{u}$ is obtained. From this vector, the $\Delta u^m_{k|k}$ elements are applied in the control system and the algorithm passes to the next iteration.

If the predicted values of output variables $y^j_{k+i|k}$ are derived by directly using a nonlinear process model then the optimization problem (1)–(4) is, in general, non–convex nonlinear optimization problem; examples of such algorithms are described e.g. in [1,5,6]. In such a Nonlinear MPC (NMPC) algorithm, different kinds of process models can be used but they are exploited in a similar way.

On the other hand, application of the MPC algorithm based on a linear model (LMPC) to obtain the prediction causes that the optimization problem (1)–(4) becomes a standard quadratic optimization problem. Unfortunately, application of an LMPC algorithm to a nonlinear process may result in unsatisfactory control performance, especially if operation in different operating points is demanded. Therefore, the fuzzy MPC (FMPC) algorithm being a combination of the LMPC and NMPC algorithms was proposed in [9]. Its formulation will be now shortly reminded. Two models are used in this algorithm. The original, nonlinear one is used to calculate the free response, whereas the fuzzy one, being a set of a few step responses, is used to calculate the dynamic matrix, updated at each algorithm iteration.

2.1 Generation of the Free Response

Let us suppose that a nonlinear process model is given (it can be practically any type of model usable in the NMPC algorithm):

$$\widehat{\boldsymbol{y}}_{k+1|k} = \boldsymbol{f}(\boldsymbol{y}_k, \boldsymbol{y}_{k-1}, \ldots, \boldsymbol{y}_{k-n_a}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_{k-2}, \ldots, \boldsymbol{u}_{k-n_b}) \ , \tag{8}$$

where $\boldsymbol{y}_{k-i} = \left[y^1_{k-i}, \ldots, y^{n_y}_{k-i}\right]^T$ is the vector of measured values of output variables at the $(k-i)^{\text{th}}$ sampling instant, $\boldsymbol{u}_{k-i} = \left[u^1_{k-i}, \ldots, u^{n_u}_{k-i}\right]^T$ is the vector of values of manipulated variables at the $(k-i)^{\text{th}}$ sampling instant; let us also denote outputs of the model at the $(k+i)^{\text{th}}$ sampling instant as $\widehat{\boldsymbol{y}}_{k+i|k} = \left[\widehat{y}^1_{k+i|k}, \ldots, \widehat{y}^{n_y}_{k+i|k}\right]$, n_a, n_b determine, how long the history of signals used by the model is.

The model (8) is then employed to obtain the free response, for the whole prediction horizon, in an iterative way, i.e.:

- First, the process model is used to obtain $\widehat{\boldsymbol{y}}_{k+1|k}$ (formula (8)).
- Then in the subsequent iterations, the values $\widehat{\boldsymbol{y}}_{k+1|k}, \ldots, \widehat{\boldsymbol{y}}_{k+i-1|k}$ are used, as the output values in the sampling instants from $(k+1)$ up to $(k+i-1)$. Moreover, as the free response is calculated, the assumption that control signal does not change ($u_{k+i} = u_{k-1}, \ i = 0, 1, ...$) is utilized. Therefore one obtains:

$$\widehat{\boldsymbol{y}}_{k+i|k} = \boldsymbol{f}(\widehat{\boldsymbol{y}}_{k+i-1|k}, \widehat{\boldsymbol{y}}_{k+i-2|k}, \ldots, \boldsymbol{y}_{k-n_a+i-1}, \boldsymbol{u}_{k-1}, \boldsymbol{u}_{k-1}, \ldots, \boldsymbol{u}_{k-n_b+i-1}) \ . \tag{9}$$

– Then, the free response is calculated taking into consideration the estimated influence of unmeasured disturbances $\boldsymbol{d}_k = \boldsymbol{y}_k - \widehat{\boldsymbol{y}}_{k|k-1}$, containing also influence of modeling errors. Thus, the final formula describing the elements of the free response is given by the formula:

$$\widetilde{\boldsymbol{y}}_{k+i|k} = \widehat{\boldsymbol{y}}_{k+i|k} + \boldsymbol{d}_k \ , \tag{10}$$

where $\widetilde{\boldsymbol{y}}_{k+i|k} = \left[\widetilde{y}^1_{k+i|k}, \ldots, \widetilde{y}^{n_y}_{k+i|k}\right]$. It is thus assumed that $\boldsymbol{d}_k$ is the same for all instants in the prediction horizon – an approach proposed in the Dynamic Matrix Control (DMC) algorithm and therefore called the DMC–type disturbance model.

2.2 Generation of the Dynamic Matrix

After calculating the free response in the way described above, the dynamic matrix, needed to predict the influence of the future control changes (generated by the algorithm) is derived using an easy to obtain Takagi–Sugeno fuzzy model. The fuzzy model has local models in the form of step responses. Therefore, its design process is very simple. It is sufficient to collect a few sets of step responses (around a few operating points). Then, using expert knowledge, the premises can be formulated and, subsequently, they can be tuned using, e.g. a fuzzy neural network. The fuzzy model is composed of the following rules:

$$\begin{aligned} &\text{Rule } f: \qquad\qquad (11)\\ &\text{if } y^{j_y}_k \text{ is } B^{f,j_y}_1 \text{ and } \ldots \text{ and } y^{j_y}_{k-n+1} \text{ is } B^{f,j_y}_n \text{ and}\\ &\quad u^{j_u}_k \text{ is } C^{f,j_u}_1 \text{ and } \ldots \text{ and } u^{j_u}_{k-m+1} \text{ is } C^{f,j_u}_m\\ &\text{then } \widehat{y}^{j,f}_k = \sum_{m=1}^{n_u} \sum_{n=1}^{p_d-1} a^{j,m,f}_n \cdot \Delta u^m_{k-n} + a^{j,m,f}_{p_d} \cdot u^m_{k-p_d} \ , \end{aligned}$$

where $y^{j_y}_k$ is the $j_y{}^{\text{th}}$ output variable value at the k^{th} sampling instant, $u^{j_u}_k$ is the $j_u{}^{\text{th}}$ manipulated variable value at the k^{th} sampling instant, $B^{f,j_y}_1, \ldots, B^{f,j_y}_n$, $C^{f,j_u}_1, \ldots, C^{f,j_u}_m$ are fuzzy sets, $a^{j,m,f}_n$ are the coefficients of step responses in the f^{th} local model, $j_y = 1, \ldots, n_y$, $j_u = 1, \ldots, n_u$, $f = 1, \ldots, l$, l is number of rules.

The output values of the fuzzy model (11) are calculated at each iteration using the following formula:

$$\widehat{y}^j_k = \sum_{m=1}^{n_u} \sum_{n=1}^{p_d-1} \widetilde{a}^{j,m}_n \cdot \Delta u^m_{k-n} + \widetilde{a}^{j,m}_{p_d} \cdot u^m_{k-p_d} \ , \tag{12}$$

where $\widetilde{a}^{j,m}_n = \sum_{f=1}^{l} \widetilde{w}_f \cdot a^{j,m,f}_n$, $\widetilde{w}_f$ are the normalized weights calculated using fuzzy reasoning, see e.g. [10,12].

The model (12) may be interpreted as the step response describing behavior of the control plant near the current operating point. This model is used to obtain at each sampling instant of the FMPC algorithm a new dynamic matrix:

$$\boldsymbol{A}_k = \begin{bmatrix} \boldsymbol{A}_k^{11} & \boldsymbol{A}_k^{12} & \dots & \boldsymbol{A}_k^{1n_u} \\ \boldsymbol{A}_k^{21} & \boldsymbol{A}_k^{22} & \dots & \boldsymbol{A}_k^{2n_u} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_k^{n_y 1} & \boldsymbol{A}_k^{n_y 2} & \dots & \boldsymbol{A}_k^{n_y n_u} \end{bmatrix}, \; \boldsymbol{A}_k^{jm} = \begin{bmatrix} \widetilde{a}_1^{j,m} & 0 & \dots & 0 & 0 \\ \widetilde{a}_2^{j,m} & \widetilde{a}_1^{j,m} & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \widetilde{a}_p^{j,m} & \widetilde{a}_{p-1}^{j,m} & \dots & \widetilde{a}_{p-s+2}^{j,m} & \widetilde{a}_{p-s+1}^{j,m} \end{bmatrix} . \tag{13}$$

2.3 Formulation of the Optimization Problem

Then, the free response (10) and the dynamic matrix (13) are used to obtain the prediction:

$$\boldsymbol{y} = \widetilde{\boldsymbol{y}} + \boldsymbol{A}_k \cdot \Delta \boldsymbol{u} \;\; , \tag{14}$$

where $\widetilde{\boldsymbol{y}} = \left[\widetilde{\boldsymbol{y}}^1, \dots, \widetilde{\boldsymbol{y}}^{n_y}\right]^T$, $\widetilde{\boldsymbol{y}}^j = \left[\widetilde{y}^j_{k+1|k}, \dots, \widetilde{y}^j_{k+p|k}\right]$. After using the prediction (14) in the performance function from (1), one obtains:

$$J_{FMPC} = (\overline{\boldsymbol{y}} - \widetilde{\boldsymbol{y}} - \boldsymbol{A}_k \cdot \Delta \boldsymbol{u})^T \cdot \boldsymbol{K} \cdot (\overline{\boldsymbol{y}} - \widetilde{\boldsymbol{y}} - \boldsymbol{A}_k \cdot \Delta \boldsymbol{u}) + \Delta \boldsymbol{u}^T \cdot \boldsymbol{\Lambda} \cdot \Delta \boldsymbol{u} \;\; , \tag{15}$$

where $\overline{\boldsymbol{y}} = \left[\overline{\boldsymbol{y}}^1, \dots, \overline{\boldsymbol{y}}^{n_y}\right]^T$, $\overline{\boldsymbol{y}}^j = \left[\overline{y}^j_k, \dots, \overline{y}^j_k\right]$, $\overline{\boldsymbol{y}}^j$ are vectors of length p,
$\boldsymbol{K} = \left[\boldsymbol{K}_1, \dots, \boldsymbol{K}_{n_y}\right] \cdot \boldsymbol{I}$; $\boldsymbol{K}_i = [\kappa_i, \dots, \kappa_i]$ have p elements,
$\boldsymbol{\Lambda} = [\boldsymbol{\Lambda}_1, \dots, \boldsymbol{\Lambda}_{n_u}] \cdot \boldsymbol{I}$; $\boldsymbol{\Lambda}_i = [\lambda_i, \dots, \lambda_i]$ have s elements.

Note that the performance function (15) depends quadratically on decision variables $\Delta \boldsymbol{u}$ and after using prediction (14) in the constraints (3) all constraints depend linearly on decision variables. Thus, one obtains a standard linear–quadratic optimization problem which is easy to solve by means of standard numerical routines.

3 Mechanisms of Disturbance Measurement Utilization

3.1 A Method Utilizing the Fuzzy Model

The first method consists in using the same approach as in the Fuzzy DMC (FDMC) algorithm detailed in [8]. This method is useful when one wants to extend the existing algorithm with the disturbance measurement utilization mechanism. Then the fuzzy process model (11) should be extended with the part which describes influence of the disturbance on the control plant. Thus, the local models have now the following form:

$$\widehat{y}_k^{j,f} = \sum_{m=1}^{n_u} \sum_{n=1}^{p_d-1} a_n^{j,m,f} \cdot \Delta u_{k-n}^m + a_{p_d}^{j,m,f} \cdot u_{k-p_d}^m + \sum_{i=1}^{p_z-1} e_i^{j,f} \cdot \Delta u_{k-i}^d + e_{p_z}^{j,f} \cdot u_{k-p_z}^d \;\; , \tag{16}$$

where $e_i^{j,f}$ $(i = 1, \ldots, p_z,\ j = 1, \ldots, n_y,\ f = 1, \ldots, l)$ are the coefficients of step responses to the change of disturbance, in the f^{th} local model, u_{k-i}^d are values of the disturbance measured in the $(k-i)^{\text{th}}$ sampling instant. Outputs of the extended fuzzy model can be thus calculated using the formula:

$$\widehat{y}_k^j = \sum_{m=1}^{n_u} \sum_{n=1}^{p_d-1} \widetilde{a}_n^{j,m} \cdot \Delta u_{k-n}^m + \widetilde{a}_{p_d}^{j,m} \cdot u_{k-p_d}^m + \sum_{i=1}^{p_z-1} \widetilde{e}_i^j \cdot \Delta u_{k-i}^d + \widetilde{e}_{p_z}^j \cdot u_{k-p_z}^d \ , \quad (17)$$

where $\widetilde{e}_n^j = \sum_{f=1}^{l} \widetilde{w}_f \cdot e_i^{j,f}$. The model (17) can be used to obtain the free response in a way as in the algorithm detailed in [8]. In such a case, a part of the free response dependent on the disturbance signal can be easily singled out; see [8]. This part of the free response can be used to extend the prediction (14):

$$\boldsymbol{y} = \widetilde{\boldsymbol{y}} + \boldsymbol{E}_k \cdot \Delta \boldsymbol{u}^d + \boldsymbol{A}_k \cdot \Delta \boldsymbol{u} \ , \quad (18)$$

where

$$\boldsymbol{E}_k = \begin{bmatrix} \boldsymbol{E}_k^1 \\ \boldsymbol{E}_k^2 \\ \vdots \\ \boldsymbol{E}_k^{n_y} \end{bmatrix}, \boldsymbol{E}_k^j = \begin{bmatrix} \widetilde{e}_1^j & \widetilde{e}_2^j - \widetilde{e}_1^j & \widetilde{e}_3^j - \widetilde{e}_2^j & \ldots & \widetilde{e}_{p_z-1}^j - \widetilde{e}_{p_z-2}^j & \widetilde{e}_{p_z}^j - \widetilde{e}_{p_z-1}^j \\ \widetilde{e}_2^j & \widetilde{e}_3^j - \widetilde{e}_1^j & \widetilde{e}_4^j - \widetilde{e}_2^j & \ldots & \widetilde{e}_{p_z}^j - \widetilde{e}_{p_z-2}^j & \widetilde{e}_{p_z}^j - \widetilde{e}_{p_z-1}^j \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \widetilde{e}_p^j & \widetilde{e}_{p+1}^j - \widetilde{e}_1^j & \widetilde{e}_{p+2}^j - \widetilde{e}_2^j & \ldots & \widetilde{e}_{p_z}^j - \widetilde{e}_{p_z-2}^j & \widetilde{e}_{p_z}^j - \widetilde{e}_{p_z-1}^j \end{bmatrix},$$

and

$$\Delta \boldsymbol{u}^d = \left[\Delta u_k^d, \Delta u_{k-1}^d, \Delta u_{k-2}^d, \ldots, \Delta u_{k-p_z+2}^d, \Delta u_{k-p_z+1}^d\right]^T \ .$$

Thus, as a result the prediction changes and the following performance function in the optimization problem (1)–(4) should be now used:

$$J_{FMPC} = (\overline{\boldsymbol{y}} - \widetilde{\boldsymbol{y}} - \boldsymbol{E}_k \cdot \Delta \boldsymbol{u}^d - \boldsymbol{A}_k \cdot \Delta \boldsymbol{u})^T \cdot \boldsymbol{K} \cdot (\overline{\boldsymbol{y}} - \widetilde{\boldsymbol{y}} - \boldsymbol{E}_k \cdot \Delta \boldsymbol{u}^d - \boldsymbol{A}_k \cdot \Delta \boldsymbol{u}) + \Delta \boldsymbol{u}^T \cdot \boldsymbol{\Lambda} \cdot \Delta \boldsymbol{u} \ . \quad (19)$$

Remark. Note that the mechanism described in this subsection can be simplified. If only one model of disturbance influence on the process is obtained, then in all local models the same values of $e_i^{j,f}$ coefficients are used, i.e. $e_i^{j,f} = e_i^j$ for $f = 1, \ldots, l$, then (17) is modified:

$$\widehat{y}_k^j = \sum_{m=1}^{n_u} \sum_{n=1}^{p_d-1} \widetilde{a}_n^{j,m} \cdot \Delta u_{k-n}^m + \widetilde{a}_{p_d}^{j,m} \cdot u_{k-p_d}^m + \sum_{i=1}^{p_z-1} e_i^j \cdot \Delta v_{k-i} + e_{p_z}^j \cdot v_{k-p_z} \ . \quad (20)$$

As the consequence, the prediction is obtained using the following formula:

$$\boldsymbol{y} = \widetilde{\boldsymbol{y}} + \boldsymbol{E} \cdot \Delta \boldsymbol{u}^d + \boldsymbol{A}_k \cdot \Delta \boldsymbol{u} \ , \quad (21)$$

where

$$\boldsymbol{E} = \begin{bmatrix} \boldsymbol{E}^1 \\ \boldsymbol{E}^2 \\ \vdots \\ \boldsymbol{E}^{n_y} \end{bmatrix}, \boldsymbol{E}^j = \begin{bmatrix} e_1^j & e_2^j - e_1^j & e_3^j - e_2^j & \ldots & e_{p_z-1}^j - e_{p_z-2}^j & e_{p_z}^j - e_{p_z-1}^j \\ e_2^j & e_3^j - e_1^j & e_4^j - e_2^j & \ldots & e_{p_z}^j - e_{p_z-2}^j & e_{p_z}^j - e_{p_z-1}^j \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ e_p^j & e_{p+1}^j - e_1^j & e_{p+2}^j - e_2^j & \ldots & e_{p_z}^j - e_{p_z-2}^j & e_{p_z}^j - e_{p_z-1}^j \end{bmatrix}.$$

Note that this time $\boldsymbol{E}$ is the matrix which is constant in each iteration and can be derived only once. Thus, calculations can be simplified.

3.2 A Method Utilizing the Nonlinear Model

The second method of disturbance measurement utilization is in fact easier to apply. Notice that if the disturbance is one of the inputs of the model used to obtain the free response, then it is sufficient to use measured values of the disturbance during calculation of the free response. If one does not have information about future values of the disturbance, then they should be assumed unchanged, like in the case of control inputs. Otherwise, estimate of future disturbance can be easily used during derivation of the free response. In the former case, the formulas given in [9] and reminded in Sect. 2.1 can be simply applied, but this time it is assumed that the vectors of inputs of the model (8) are extended:

$$\widetilde{\boldsymbol{u}}_{k-i} = \left[u^1_{k-i}, \ldots, u^{n_u}_{k-i}, u^d_{k-i}\right]^T \quad , \tag{22}$$

where u^d_{k-i} is value of disturbance in the $(k-i)^{th}$ sampling instant. Thus, the nonlinear process model is now given by:

$$\widehat{\boldsymbol{y}}_{k+1|k} = \boldsymbol{f}(\boldsymbol{y}_k, \boldsymbol{y}_{k-1}, \ldots, \boldsymbol{y}_{k-n_a}, \widetilde{\boldsymbol{u}}_{k-1}, \widetilde{\boldsymbol{u}}_{k-2}, \ldots, \widetilde{\boldsymbol{u}}_{k-n_b}) \ , \tag{23}$$

and the elements of the free response – by:

$$\widehat{\boldsymbol{y}}_{k+i|k} = \boldsymbol{f}(\widehat{\boldsymbol{y}}_{k+i-1|k}, \widehat{\boldsymbol{y}}_{k+i-2|k}, \ldots, \boldsymbol{y}_{k-n_a+i-1}, \widetilde{\boldsymbol{u}}_{k-1}, \widetilde{\boldsymbol{u}}_{k-1}, \ldots, \widetilde{\boldsymbol{u}}_{k-n_b+i-1}) \ . \tag{24}$$

4 Simulation Experiments

The methods of disturbance measurement utilization were tested in the control system of the isothermal CSTR in which the van de Vusse reaction carries out [4], used for tests also in [9]. The process model of the reactor contains two composition balance equations

$$\begin{array}{c} \frac{dC_\mathrm{A}}{dt} = -k_1 \cdot C_\mathrm{A} - k_3 \cdot C_\mathrm{A}^2 + \frac{F}{V}\left(C_\mathrm{Af} - C_\mathrm{A}\right) \ , \\ \frac{dC_\mathrm{B}}{dt} = k_1 \cdot C_\mathrm{A} - k_2 \cdot C_\mathrm{B} - \frac{F}{V} C_\mathrm{B} \ , \end{array} \tag{25}$$

where C_A, C_B are the concentrations of components A and B, respectively, F is the inlet flow rate (equal to the outlet flow rate), V is the volume in which the reaction takes place (it is assumed constant and $V = 1$ l), C_Af is the concentration of component A in the inlet flow stream (it is assumed that $C_\mathrm{Af} = 10$ mol/l). The values of parameters are: $k_1 = 50$ 1/h, $k_2 = 100$ 1/h, $k_3 = 10$ l/(h $\cdot$ mol). The output variable is the concentration C_B of substance B, the manipulated variable is the inlet flow rate F of the raw substance, and C_Af concentration is the disturbance variable.

The FMPC controller used in [9] was extended with the mechanisms of disturbance measurement utilization. Thus, during the experiments the sampling period was assumed equal to $T_s = 3.6$ s; tuning parameters were assumed as follows: prediction horizon $p = 70$, control horizon $s = 35$, weighting coefficient $\lambda = 0.001$. The fuzzy model used in the FMPC algorithm, for derivation of the dynamic matrix, is composed of three step responses obtained near the following operating points:

P1) $C_{B0} = 0.91$ mol/l, $C_{A0} = 2.18$ mol/l, $F_0 = 20$ l/h;
P2) $C_{B0} = 1.12$ mol/l, $C_{A0} = 3$ mol/l, $F_0 = 34.3$ l/h;
P3) $C_{B0} = 1.22$ mol/l, $C_{A0} = 3.66$ mol/l, $F_0 = 50$ l/h.

The membership functions, assumed after analysis of the steady–state characteristics of the control plant, are shown in Fig. 1.

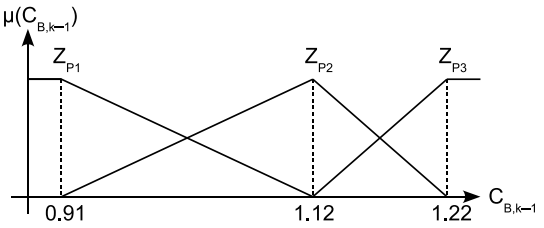


Fig. 1. Membership functions of the fuzzy model used in the FMPC controller

During the experiments the operation of the control system with the FMPC algorithm extended with mechanisms of disturbance measurement was tested. The responses to the change of the disturbance C_{Af} by 10% from $C_{Af0} = 10$ mol/l to $C_{Af1} = 11$ mol/l were obtained. The disturbance changed at the 6^{th} minute of simulation. In the case of the method based on the fuzzy model its simplified version was used.

First, the experiment near the set–point value $\overline{C}_B = 1.02$ was done. In the case of both mechanisms of disturbance measurement utilization significant improvement of the responses was obtained comparing to the case when none of the mechanisms was used (dotted lines in Fig. 2). The maximum control error is around two times smaller when any of the two mechanisms is used. The mechanism based on the nonlinear model (solid lines in Fig. 2) works better than the one based on the fuzzy model (dashed lines in Fig. 2) – the output value achieves the set–point faster (before the 8^{th} minute), and changes of the control signal are smaller. Moreover, in the case of the method based on the fuzzy model oscillations occur. The next experiment, was done near the set–point value $\overline{C}_B = 1.22$. This time, both methods also prove to be effective but both give very similar result. It is because the simplified model of the influence of the disturbance on the control plant reflects the real dependencies in the control plant near the current operating point well. It illustrates the importance of the

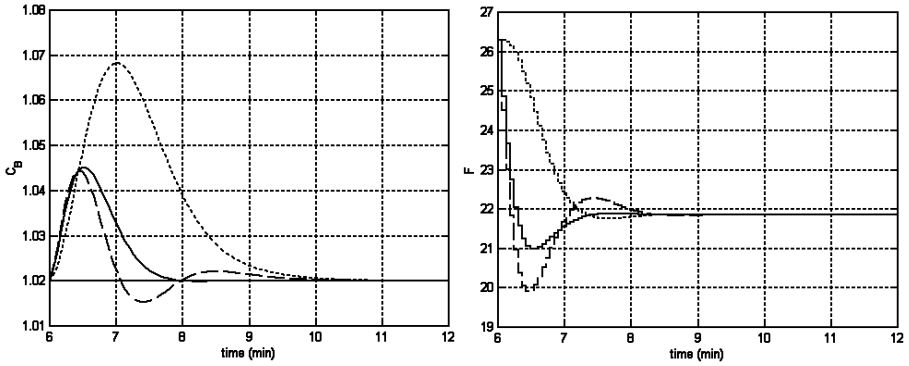


Fig. 2. Responses of the control system to the change of the disturbance by 10% to $C_{\mathrm{Af1}} = 11$ mol/l near the set–point value $\overline{C}_{\mathrm{B}} = 1.02$; mechanism of disturbance measurement utilization: not used – dotted lines, based on the fuzzy model – dashed lines, based on the nonlinear model – solid lines

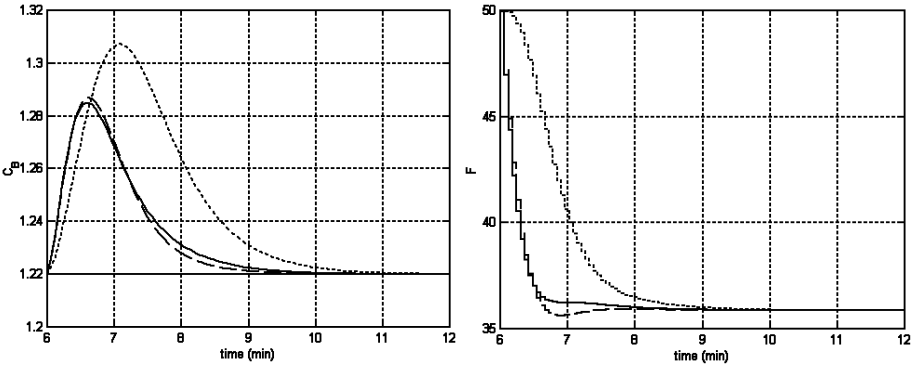


Fig. 3. Responses of the control system to the change of the disturbance by 10% to $C_{\mathrm{Af1}} = 11$ mol/l near the set–point value $\overline{C}_{\mathrm{B}} = 1.22$; mechanism of disturbance measurement utilization: not used – dotted lines, based on the fuzzy model – dashed lines, based on the nonlinear model – solid lines

quality of the model for the control performance and capabilities lurking in the method based on the fuzzy model.

5 Summary

The mechanisms of disturbance measurement utilization added to the efficient predictive algorithm based on fuzzy and nonlinear models are discussed in the paper. The first mechanism can be used to extend capabilities of the existing algorithm implementation. It utilizes a relatively easy–to–obtain fuzzy model of

influence of a disturbance on the control plant. The second mechanism can give better results than the first one because it exploits a nonlinear model used to obtain the free response. The algorithm is modified in such a way that its numerical efficiency is maintained. It is because the mechanisms consist in modifications affecting only the free response. Therefore, the control signal is still calculated as a solution of the quadratic optimization problem. Example experiments show that the discussed mechanisms of disturbance measurement utilization, though simple, can bring significant improvement of control performance.

References

1. Babuska, R., te Braake, H.A.B., van Can, H.J.L., Krijgsman, A.J., Verbruggen, H.B.: Comparison of intelligent control schemes for real–time pressure control. Control Engineering Practice 4, 1585–1592 (1996)
2. Blevins, T.L., McMillan, G.K., Wojsznis, W.K., Brown, M.W.: Advanced Control Unleashed. ISA (2003)
3. Camacho, E.F., Bordons, C.: Model Predictive Control. Springer (1999)
4. Doyle, F., Ogunnaike, B.A., Pearson, R.K.: Nonlinear model–based control using second–order Volterra models. Automatica 31, 697–714 (1995)
5. Fink, A., Fischer, M., Nelles, O., Isermann, R.: Supervision of nonlinear adaptive controllers based on fuzzy models. Control Engineering Practice 8, 1093–1105 (2000)
6. Foss, B.A., Johansen, T.A., Sorensen, A.V.: Nonlinear predictive control using local models–applied to a batch fermentation process. Control Engineering Practice 3, 389–396 (1995)
7. Maciejowski, J.M.: Predictive control with constraints. Prentice Hall, Harlow (2002)
8. Marusak, P.: Advantages of an easy to design fuzzy predictive algorithm in control systems of nonlinear chemical reactors. Applied Soft Computing 9, 1111–1125 (2009)
9. Marusak, P.M.: Efficient Model Predictive Control Algorithm with Fuzzy Approximations of Nonlinear Models. In: Kolehmainen, M., Toivanen, P., Beliczynski, B. (eds.) ICANNGA 2009. LNCS, vol. 5495, pp. 448–457. Springer, Heidelberg (2009)
10. Piegat, A.: Fuzzy Modeling and Control. Physica–Verlag, Berlin (2001)
11. Rossiter, J.A.: Model-Based Predictive Control. CRC Press, Boca Raton (2003)
12. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. IEEE Trans. Systems, Man and Cybernetics 15, 116–132 (1985)
13. Tatjewski, P.: Advanced Control of Industrial Processes; Structures and Algorithms. Springer, London (2007)

Fast Submanifold Learning with Unsupervised Nearest Neighbors

Oliver Kramer

Computational Intelligence Group
Department of Computing Science
University of Oldenburg
oliver.kramer@uni-oldenburg.de

Abstract. We present an unsupervised nearest neighbors (UNN) variant for continuous latent spaces that allows to embed patterns in different submanifolds. The problem to simultaneously assign patterns to models and learn the embeddings can be very challenging, as the manifolds may lie closely to each other and can have different dimensions and arbitrary curvature. The UNN-based submanifold learning approach (SL-UNN) that is proposed in this paper combines a fast constructive K-means variant with the UNN manifold learning approach. The resulting speedy approach depends on only few parameters, i.e., a distance threshold to allow the definition of new clusters and the usual UNN parameters. Extensions of SL-UNN are able to automatically determine parameter of each submanifold based on the data space reconstruction error.

1 Introduction

Efficient and robust dimensionality reduction methods are required to process high-dimensional patterns, e.g., for visualization, as preprocessing for classification and other methods like symbolic algorithms. With increasing data sets and improved sensor systems, dimensionality reduction becomes an important problem class in machine learning. Dimensionality reduction methods perform a mapping $\mathbf{F} : \mathbb{R}^d \to \mathbb{R}^q$ from a high-dimensional data space $\mathbb{R}^d$ to a latent space of lower dimensionality $\mathbb{R}^q$ with $q < d$. The goal is that the latent representations maintain neighborhoods and distances of the high-dimensional data patterns.

Patterns may lie in different submanifolds. To allow the simultaneous assignment to clusters and learning of submanifolds, we introduce a submanifold variant of KUNN in this section. Before we introduce the UNN-based variant for submanifold learning, we describe the manifold clustering problem and present related work in this area. In this paper we present an extension of kernel unsupervised nearest neighbors (KUNN) [4] to submanifold learning. In KUNN latent points are iteratively and stochastically embedded. Distances in data space are employed as standard deviation for Gaussian sampling in latent space, neighborhoods are preserved with the KNN-based data space reconstruction error. A kernel function implicitly maps patterns to a feature space of higher dimensionality, where patterns are potentially linearly separable.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 317–325, 2013.

1.1 Submanifold Learning

Given a matrix of N patterns $\mathbf{Y} = [\mathbf{y}_i]_{i=1}^N \in \mathbb{R}^{d \times N}$ lying in τ different manifolds $\{\mathcal{M}_j\}_{j=1}^\tau$ with intrinsic dimensions $\{d_j\}_{j=1}^\tau$, the submanifold learning problem is the task to simultaneously assign the patterns to clusters and solve the manifold learning problem independently within each cluster. We seek for a low-dimensional representation, i.e., a matrix of latent points $\mathbf{X} = [\mathbf{x}_i]_{i=1}^N$, such that a regression function $\mathbf{f}$ applied to matrix $\mathbf{X}$ *point-wise optimally reconstructs the patterns* $\mathbf{Y}$, i.e., $\mathbf{f}$ maps from $\mathbb{R}^{q \times N}$ to $\mathbb{R}^{d \times N}$. The optimization problem can be formalized as follows

$$\text{minimize } E(\mathbf{X}) := \frac{1}{N} \|\mathbf{Y} - \mathbf{f}(\mathbf{X})\|_F^2 \tag{1}$$

with Frobenius norm $\|\cdot\|_F^2$. Error $E(\mathbf{X})$ is called data space reconstruction error (DSRE). The latent points $\mathbf{X}$ define the low-dimensional representation. The regression function $\mathbf{f}$ applied to the latent points should optimally *reconstruct* the high-dimensional patterns and induces its characteristic to the learning result.

The problem to simultaneously learn submanifolds and their embeddings is difficult to solve. First, clusters have to be identified, second low-dimensional representations of the patterns have to be learned. Further, in each cluster possibly varying parameters may be necessary. Vidal [9] summarizes challenges of the submanifold learning problem. An essential characteristic of submanifold learning is the coupling between segmentation of patterns and model estimation. A known segmentation would simplify the model estimation process, as it would be clear, which patterns belong to which manifold. In turn, a known distribution would simplify the estimation process, as the model would allow to determine the assignment to manifolds. In general, the distribution within the clusters is unknown. Furthermore, closeness between subspaces, or intersections extremely complicate the decomposition and the model estimation process. The perspective of the data space as collection of submanifolds with varying characteristics is similar to the concept of local models that allow separate parameterizations. In [5] we proposed a kernel regression approach with multiple local models and independent kernel parameters.

1.2 Related Work

Numerous submanifold learning algorithms have been presented in the past. Algebraic methods are based on matrix factorization [2,3] and polynomial algebra, e.g., fitting polynomials to the submanifolds [10]. Iterative methods are often extensions of K-means, alternately fitting PCA-models to each submanifold and then assigning each pattern to its closest submanifold. K-planes [1] and K-subspaces [8] are instances of these algorithms. To handle noise, statistical models like mixtures of probabilistic PCA [7] assume that data within submanifolds are generated with independent Gaussian distributions employing the maximum likelihood principle. The assumption of Gaussian mixtures is the basis of the agglomerative lossy compression approach [6], which minimizes coding length required for fitting the Gaussians.

2 Submanifold Learning UNN

2.1 Constructive K-Means

A fast variant of K-means is an iterative algorithm that does not require the initial specification of the number of clusters. To determine if new patterns belong to a novel cluster, a distance threshold $\zeta \in \mathbb{R}^+$ must be defined. The idea of the fast constructive K-means variant is to iteratively assign each pattern to the cluster of the closest codebook vector and then to update the center. The first pattern $\mathbf{y}_1$ becomes the first cluster center $\mathbf{c}_1 = \mathbf{y}_1$, and we set $\mathcal{I}(\mathbf{y}_1, \mathbf{c}_1) = 1$.

Let n be the number of patterns that have been assigned to clusters. For all remaining patterns $\mathbf{y}_i$ with $i = n + 1 \leq N$, the algorithm looks for the closest codebook vector $\mathbf{c}^*$. If the distance to pattern $\mathbf{y}_i$ exceeds a threshold $\|\mathbf{y}_i - \mathbf{c}^*\|^2 > \zeta$, it becomes a new cluster center. We increase the number of clusters $K = K+1$ and set $\mathbf{c}_K = \mathbf{y}_i$, as well as $\mathcal{I}(\mathbf{y}_i, \mathbf{c}_K) = 1$. Otherwise, pattern $\mathbf{y}_i$ is assigned to the cluster of codebook vector $\mathbf{c}^*$, i.e., we set $\mathcal{I}(\mathbf{y}_i, \mathbf{c}^*) = 1$. Further, the cluster has to be updated

$$\mathbf{c}^* = \frac{\sum_{i=1}^{n+1} \mathcal{I}(\mathbf{y}_i, \mathbf{c}^*) \cdot \mathbf{y}_i}{\sum_{i=1}^{n+1} \mathcal{I}(\mathbf{y}_i, \mathbf{c}^*)}. \tag{2}$$

This process is repeated until all patterns are assigned to clusters. The result of the iterative method depends on the order of elements. The algorithm requires the specification of threshold ζ to determine, if it is reasonable to start a new cluster. The clustering result is very sensitive to this parameter. The experimental analysis in Section 3 will show that too large values result in too few clusters, while too small values generate too many small clusters. For $\zeta \leq \min \|\mathbf{y}_i - \mathbf{y}_j\|^2$ with $i, j = 1, \ldots, N$ and $i \neq j$, constructive K-means returns N submanifolds.

2.2 SL-UNN

The new submanifold learning approach SL-UNN is based on the idea to combine constructive K-means with UNN. The iterative construction scheme of K-means can easily be combined with the iterative UNN scheme. A pattern is assigned to a submanifold $\mathcal{M}_j$ and immediately embedded in $\mathcal{M}_j$. Algorithm 1.1 shows the pseudocode of SL-UNN.

The first pattern $\mathbf{y}_1$ is assigned to the first manifold and embedded at an initial latent position, i.e., $\mathcal{M}_1 = \{(\mathbf{0}, \mathbf{y}_1)\}$. Let n be the number of patterns that have been assigned to submanifolds. For all remaining patterns $\mathbf{y}_i$ with $i = n + 1 \leq N$, constructive K-means assigns $\mathbf{y}_i$ to a manifold $\mathcal{M}_j$, where the pattern is embedded with UNN. An arbitrary latent position $\mathbf{x}_i$ can be chosen, when a new manifold is started. In the experimental part, we will place the latent submanifold centers on a lattice with reasonable distances between the grid points that allow to distinguish between different manifolds.

The embedding is only based on the patterns that are assigned to $\mathcal{M}_j$, i.e., only the patterns of $\mathcal{M}_j$ are used for the neighborhood search and the DSRE

computation, while patterns that are not part of the submanifold are neglected. For each manifold, separate parameterizations allow to improve the model estimations. In the experimental part, we will allow to use different kernel functions with separate parameters.

Algorithm 1.1. SUBMANIFOLD LEARNING UNN

Require: $\mathbf{Y}$, ζ
1: $K = 1$
2: $\mathbf{c}_1 = \mathbf{y}_1$
3: $\mathcal{M}_1 = \{(\mathbf{0}, \mathbf{y}_1)\}$
4: **for** $i = 2$ **to** N **do**
5: constructive K-means: j is index of last cluster
6: embed $\mathbf{y}_i$ in $\mathcal{M}_j$ with UNN and $\kappa, \sigma_j^* \rightarrow \mathbf{x}_i$
7: $\mathcal{M}_j = \mathcal{M}_j \bigcup \{(\mathbf{x}_i, \mathbf{y}_i)\}$
8: **if** $|\mathcal{M}_j| = \vartheta$ **then**
9: choose parameters σ_j^* for manifold $\mathcal{M}_j$
10: **end if**
11: **end for**

The runtime complexity of the approach lies in the same class as UNN. SL-UNN iteratively assigns a pattern to the closest codebook vector, whose number is upper bounded by τ, in $O(\tau) = O(1)$ steps and embeds a pattern in the assigned manifold $\mathcal{M}_j$ in $O(\log N)$ resulting in an overall runtime of $O(N \cdot \tau + N \log N) = O(N \log N)$.

The parameters in each manifold can be adapted to allow better manifold learning results. For this purpose, we introduce the following approach. When ϑ patterns have been embedded in submanifold $\mathcal{M}_j$ with $j = 1, \ldots, \tau$, the algorithm optimizes the corresponding settings, which can be dimensionalities $\{d_j\}_{j=1}^{\tau}$, kernel functions $\{k\}_{j=1}^{\tau}$, and corresponding kernel parameters. The two most important variants for the parameter adaptation process are: (1) the -definition of parameter sets $\{\sigma\}_{f=1}^{g}$ that are successively tested and (2) optimization of parameters within defined bounds. We will employ the first variant in the following experimental analysis. The latter is recommendable for parameter sets that are too large to be enumerated completely.

3 Experimental Analysis

In this section, we analyze SL-UNN experimentally on the typical data sets of the previous experiments and compare to KUNN without manifold clustering.

3.1 Digits

The first experimental part focuses on the visualization of the submanifold learning results on the *Digits* data set. We expect that SL-UNN identifies different

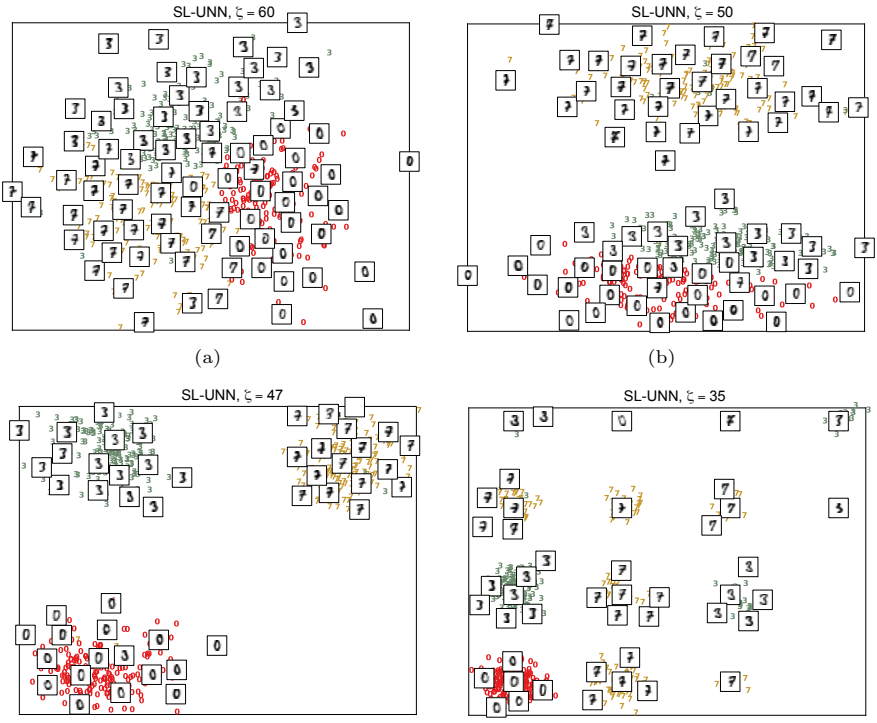


Fig. 1. Comparison of SL-UNN embeddings of the *Digits* data set ('0', '3', and '7') in 2-dimensional manifolds with varying ζ

clusters corresponding to the different classes of *Digits*, and embeds the patterns within each cluster. As first data set, we consider an example consisting of three classes. Figure 1 shows the results of SL-UNN with various settings for ζ on the *Digits* data set with $N = 539$ patterns and digits '0', '3', and '7'. We choose $\kappa = 20$ and neighborhood size $K = 10$. For a too large threshold $\zeta = 60.0$, only one manifold has been found. Hence, the result corresponds to standard UNN without clustering. For $\zeta = 50.0$, two manifolds have been identified. In particular, the '0's are separated from the rest of the patterns with exception of few outliers. For $\zeta = 47.0$, the number of manifolds is identical with the number of classes, i.e., $\tau = 3$. This is the optimal setting, which reflects the different classes and corresponding reasonable embeddings. However, we can observe few false patterns in some manifolds. This is caused by errors of the constructive K-means clustering procedure. For too small threshold values, too many manifolds are found, e.g., $\tau = 21$ manifolds in case of $\zeta = 35.0$.

For a further data set composed of the *Digits* data set with six classes (digits from '0' to '5'), we can observe similar results, see Figure 2. For too large distance thresholds like $\zeta = 47.0$, only two clusters were found, while for $\zeta = 45.0$ already four clusters have been detected and even $\tau = 25$ in case of $\zeta = 35.0$. But with

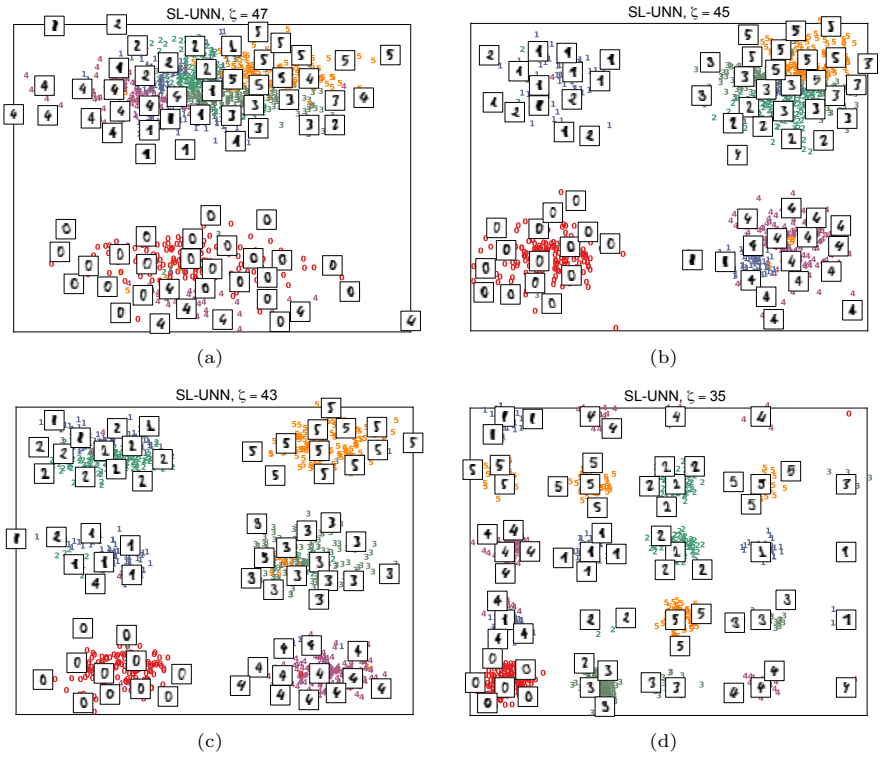


Fig. 2. Comparison of SL-UNN embeddings of the *Digits* data set with six classes ('0' to '5') in 2-dimensional manifolds with varying ζ

the setting $\zeta = 43.0$, SL-UNN identifies a number of manifolds that corresponds to the number of classes.

3.2 DSRE in Submanifolds

In the following, we analyze the DSRE within the submanifolds. The parameters for the model of each manifold are adapted according to the following scheme after ϑ patterns have been assigned to manifold $\mathcal{M}_j$:

1. linear kernel,
2. polynomial kernel with $p = 2$,
3. RBF-kernel with parameters $\gamma = 10^{2l}, 0 \leq l \leq 4$,
4. hyperbolic tangent kernel with $a = 10^{-l}$, $1 \leq l \leq 6$, and $b = -10^{-2}$.

For each manifold $\{\mathcal{M}_j\}_{j=1}^{\tau}$, the setting $\{\sigma_j^*\}_{j=1}^{\tau}$ that achieves a minimal DSRE with $\vartheta = 50$ patterns is chosen.

Table 1 shows the experimental comparison between SL-UNN with $\zeta = 43.0$ and KUNN without manifold clustering in 25 runs. The figures show the

Table 1. Analysis of overall DSRE* and submanifold DSRE* within $\{\mathcal{M}_j\}_{j=1}^{\tau}$ for the *Digits* data set with neighborhood sizes $K = 5, 10, 30$, $\kappa = 30$, and $\zeta = 43.0$.

Digits	$\mathcal{M}_1$	$\mathcal{M}_2$	$\mathcal{M}_3$	$\mathcal{M}_4$	$\mathcal{M}_5$	$\mathcal{M}_6$	SL-UNN (1,083)	KUNN (180)	KUNN (1,083)
k	DSRE*						DSRE*	DSRE*	DSRE*
5	2.53	8.12	22.39	6.22	11.86	17.64	14.49 ± 0.12	16.16 ± 0.25	13.83 ± 0.11
10	3.06	9.96	27.41	7.43	13.97	20.46	17.51 ± 0.30	20.94 ± 0.50	16.48 ± 0.065
30	3.63	10.94	35.86	9.33	17.74	23.52	22.44 ± 0.34	28.76 ± 1.28	22.71 ± 1.39

average pattern-wise DSRE* $E(\mathbf{X}) = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\|\mathbf{f}(\mathbf{x}_i) - \mathbf{y}_i\|^2}$, which is not decreasing with an increasing number of patterns. This measure is reasonable as the submanifolds contain less patterns than the overall manifold. The results are compared to the DSRE of the submanifolds $\{\mathcal{M}_j\}_{j=1}^{\tau}$. In Table 1, the average DSRE* in each manifold is presented, as well as the DSRE* of the whole embedding. We compare SL-UNN with $\zeta = 43.0$ and $\kappa = 30$ on $N = 1,083$ patterns to KUNN on $N = 180$ patterns with one manifold ($\tau = 1$) corresponding to the average number of patterns in one manifold ($N = 180 \approx 1080/6$). Further, we compare to KUNN (1,083) with $\tau = 1$ employing all $N = 1,083$ patterns.

The results show that SL-UNN assigns the patterns to six submanifolds. In four of the six submanifolds, significantly lower errors are achieved than (1) the average DSRE* of the whole embedding and than (2) KUNN with $N = 180$ patterns (which corresponds to the average number of patterns in each submanifold). The overall DSRE* is slightly higher than the overall DSRE* of KUNN on the whole data set. The reason is that the submanifolds are placed on discrete grid positions not taking into account a sorting w.r.t. the DSRE. Hence, the neighborhood preservation for patterns in different submanifolds can be violated. A modification of SL-UNN could be to *sort* the clusters w.r.t. the DSRE when a new cluster is started. SL-UNN chose the hyperbolic tangent kernel in four of the submanifolds, while the kernel function choice for the other two manifolds varied from RBF-kernel with different settings to the polynomial kernel. The KUNN variants chose the hyperbolic tangent kernel in each experiment.

3.3 ISOMAP-Faces

The embedding results of SL-UNN on the ISOMAP-*Faces* data set are shown in Figure 3. The figures show the embeddings of SL-UNN with two parameter settings, i.e., $\zeta = 20.0$ and $\zeta = 17.0$. In all experiments, the settings $\kappa = 20$ and neighborhood size $K = 5$ have been chosen. In contrast to the *Digits* data set, no labels are known in advance, and we cannot draw conclusions about the correct number of manifolds. But we can observe that patterns embedded in each manifold have similar characteristics, i.e., similar poses and similar lights. Further, within the manifolds neighbored patterns are neighbored in latent space. Both observations underline that SL-UNN learns reasonable submanifolds.

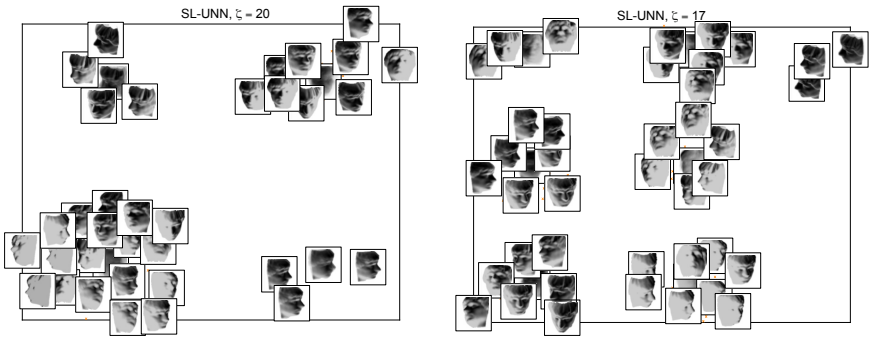


Fig. 3. SL-UNN embeddings of the ISOMAP-*Faces* data set with the two parameters $\zeta = 20.0$ and $\zeta = 17.0$

4 Conclusions

Patterns may lie in different submanifolds with varying characteristics. To allow the assignment of patterns to submanifolds, we have extended UNN by a constructive clustering approach. The novel algorithm called SL-UNN assigns patterns to submanifolds based on a simple yet efficient K-means variant and simultaneously embeds them with UNN. Submanifolds allow the management of separate parameterizations, e.g., kernel functions and kernel parameters in case of KUNN. The overall runtime complexity is still $O(N \log N)$, if space partitioning data structures offer neighborhood requests in $O(\log N)$. The experimental results have shown that SL-UNN achieves low learning errors in the majority of the submanifolds, as each local model can employ separate parameters. Further, a speedup can be observed, as the submanifolds, where the embeddings take place, are smaller than the whole data set. If low errors are required in local neighborhoods, the employment of SL-UNN can be recommended. However, the overall DSRE could not be decreased. The clustering mechanism can be extended by further clustering and manifold criteria that may be appropriate for specific scenarios, e.g., by density-based measures. The prize of higher clustering accuracies and possibly lower embedding errors might be paid with worse runtime complexities.

References

1. Bradley, P.S., Mangasarian, O.L.: k-plane clustering. Journal of Global Optimization 16, 23–32 (2000)
2. Costeira, J.P., Kanade, T.: A multibody factorization method for independently moving objects. International Journal of Computer Vision 29(3), 159–179 (1998)
3. Gear, C.W.: Multibody grouping from motion images. International Journal of Computer Vision 29(2), 133–150 (1998)

4. Kramer, O.: Unsupervised Nearest Neighbors with Kernels. In: Glimm, B., Krüger, A. (eds.) KI 2012. LNCS, vol. 7526, pp. 97–106. Springer, Heidelberg (2012)
5. Kramer, O., Satzger, B., Lässig, J.: Power Prediction in Smart Grids with Evolutionary Local Kernel Regression. In: Graña Romay, M., Corchado, E., Garcia Sebastian, M.T. (eds.) HAIS 2010, Part I. LNCS, vol. 6076, pp. 262–269. Springer, Heidelberg (2010)
6. Ma, Y., Derksen, H., Hong, W., Wright, J.: Segmentation of multivariate mixed data via lossy data coding and compression. IEEE Transaction on Pattern Analysis and Machine Intelligence 29(9), 1546–1562 (2007)
7. Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analysers. Neural Computation 11(2), 443–482 (1999)
8. Tseng, P.: Nearest q-flat to m points. Journal of Optimization Theory and Applications 105, 249–252 (2000)
9. Vidal, R.: Subspace clustering. IEEE Signal Processing Magazine 28(2), 52–68 (2011)
10. Vidal, R., Ma, Y., Sastry, S.: Generalized principal component analysis (gpca). IEEE Transactions on Pattern Analysis and Machine Intelligence 27(12), 1945–1959 (2005)

Using Carrillo-Lipman Approach to Speed up Simultaneous Alignment and Folding of RNA Sequences

Mária Šimaľová

Institute of Computer Science
Pavol Jozef Šafárik University in Košice
Jesenná 5, Košice, Slovak Republic
maria.simalova@gmail.com

Abstract. Multiple sequence alignment and RNA folding are two important tasks in the field of bioinformatics. Solving those problems simultaneously leads to biologically more significant results. The only one currently known precise algorithm (Sankoff) is too much computationaly expensive for such long sequences. In this work we introduce a new algorithm, that is a combination of well known Nussinov folding algorithm and Sankoff quadratic alignment algorithm and a speed-up for this algorithm that is inspired by the Carrillo-Lipman algorithm for the multiple sequence alignment problem. This approach may allow us to simultaneously align and fold more than two sequences in a better time than we can do it using the Sankoff algorithm.

Keywords: multiple sequence alignment, RNA folding.

1 Multiple Sequence Alignment

Multiple sequence alignments are an essential tool for protein structure and function prediction, phylogeny inference and other common tasks in sequence analysis. The computation of multiple sequence alignment is not a trivial task, because it is hard to precisely define the properties of biologically optimal alignment. The standard computational formulation of the pairwise problem is to identify the alignment that maximizes sequence similarity, which is typically defined as the sum of substitution matrix scores for each aligned pair of residues, minus some penalties for gaps. This approach is generalized to the multiple sequence case by seeking an alignment that maximizes the sum of similarities for all pairs of sequences (the sum-of pairs, or SP, score)[1].

2 Dynamic Programming

Algorithms based on the SP scores produce a mathematically, but not necessarily biologically exact alignment. Wang & Jiang in 1994 [2] showed that the minimal time and memory required to find the alignment with maximal SP score

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 326–335, 2013.

grows exponentially with growing number of sequences. The basic dynamic programming algorithm finding the pairwise sequence alignment(alignment of two sequences) from Needleman & Wunsch [3] fills in a two dimensional table D. Let R_1 and R_2 be two sequences, that we want to align. The value of $D[i,j]$ is the maximal possible value of aligning first i bases of R_1 with first j bases of R_2. There are 3 different ways to fill in the value $D[i,j]$:

- Align base i of R_1 with space. In this case we have to add the space penalty y to the alignment of first $i-1$ bases of R_1 and first j bases of R_2.
- Align base j of R_2 with space. In this case we have to analogically add the space penalty y to the alignment of first i bases of R_1 and first $j-1$ bases of R_2.
- Align base i of R_1 with base j of R_2. The value of $D[i,j]$ will be the sum of the alignment value of first $i-1$ bases of R_1 and $j-1$ bases of R_2 and the value of aligning base i with base j ($C[i,j]$).

$$D[i,j] = max \begin{cases} D[i-1,j-1] + C(R_{1,i}, R_{2,j}) \\ D[i-1,j] + y \\ D[i,j-1] + y \end{cases} \tag{1}$$

$$D[i,0] = i * y, D[0,j] = j * y \tag{2}$$

The matrix is then filled using equation (1) and the initial condition (2). The time and space complexity of Needleman-Wunsch algorithm is $O(n^2)$.

This algorithm can be generalized for N sequences (3), but the number of cases is growing up exponentially with growing number of sequences.

$$D[\overrightarrow{i}] = max_{p \in Perm_N \{0,1\}} \begin{cases} D[\overrightarrow{i-p}] + \sum_{j=1}^{N} p_j * (N - \sum_{j=1}^{N} p_j) * y \\ \quad + \sum_{m,l \in \{1,\ldots,N\}, p_m = p_l = 1} C(R_{m,i_m}, R_{l,i_l}) \end{cases} \tag{3}$$

The space complexity of generalized algorithm is $O(n^N)$ and the time complexity is $O(2^N * N^2 * n^N)$.

2.1 Carrillo-Lipman

The idea of Carrillo-Lipman heuristic [4] is to search only among those alignments whose value is higher then some constant U. If we set U to be the value of some unoptimal alignment found by some faster heuristic alignment algorithm, then the result will be optimal. The main idea is to count the matrix values in a different way. After improving cell $D[i_1, i_2, ..., i_N]$ improve all cells that can be affected by this change only if their value will be "high enough".

Lets take a look about what it means "high enough". If we count the value of optimal pairwise alignment for all pairs of sequences such that $d_{m,l}[i_m, i_l]$ will be the maximal value for aligning subsequences $R_m[i_m, ..., n_m]$ and $R_l[i_l, ..., n_l]$, then it is obvious that:

$$D[\overrightarrow{n}] - \sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}[i_m, i_l] \leq D[\overrightarrow{i}] \tag{4}$$

Suppose we know some possible value of searched multiple alignment U. We can exclude the cell $D[i_1, i_2, ..., i_N]$ from the computation if its value is so small that even if the alignment value of subsequences $R_1[i_1, ..., n_1]$, $R_2[i_2, ..., n_2]$,..., $R_N[i_N, ..., n_N]$ is maximal($\sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}[i_m, i_l]$), then the whole alignment value will be less than U:

$$D[\overrightarrow{i}] + \sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}[i_m, i_l] < U \tag{5}$$

2.2 Sankoff Quadratic Alignment

Sankoff quadratic pairwise alignment is based on comupting matrix of size $O(n^4)$. The value of $D[i, j; k, l]$ is the maximal value for aligning subsequences $R[i, ..., j]$ with $R'[k, ..., l]$. This method is computationally worse then Needleman-Wunsch algorithm, but we will need it later. The equations for filling it in are:

$$D[i, j; k, l] = max \begin{cases} D[i, j; k, l-1] + y \\ D[i, j-1; k, l] + y \\ D[i, j-1; k, l-1] + C(R_j, R'_k) \end{cases} \tag{6}$$

$$D[i, i; k, k] = C(i, k), D[0, j; 0, 0] = j * y, D[0, 0; 0, k] = k * y \tag{7}$$

Sankoff quadratic alignment algorithm can be generalized for N sequences as well as Needleman-Wunsch algorithm(equation (8)). Its space complexity will be $O(n^{2*N})$ a needs $O(n^{2*N} * 2^N * N^2)$ time.

$$D[\overrightarrow{i, j}] = max_{p \in Perm_N\{0,1\}} \begin{cases} D[\overrightarrow{i, j-p}] + \sum_{k=1}^{N} p_k * (N - \sum_{k=1}^{N} p_k) * y \\ \quad + \sum_{m,l \in \{1,...,N\}, p_m = p_l = 1} C(R_{m,i_m}, R_{l,i_l}) \end{cases} \tag{8}$$

2.3 Applying Carrillo-Lipman Approach to Sankoff Multiple Alignment

If we want to apply the Carrillo-Lipman approach to the last mentioned algorithm, we need to divide the alignment to three parts instead of two. So we have to precompute $d_{m,l}[i_m, i_l]$ that will be the maximal possible pairwise alignment value of subsequences $R_m[0, ..., i_m]$ and $R_l[0, ..., i_l]$, and $d'_{m,l}[j_m, j_l]$ will be the maximal possible pairwise alignment value of subsequences $R_m[j_m, ..., n_m]$ and $R_l[j_l, ..., n_l]$. Analogically to equation (4), we can see, that:

$$D[\overrightarrow{0, n}] - \sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}[i_m, i_l] - \sum_{m=1}^{N} \sum_{l=1}^{N} d'_{m,l}[j_m, j_l] \leq D[\overrightarrow{i, j}] \tag{9}$$

The cell $D[\overrightarrow{i, j}]$ can then be excluded from the computation if its value is so small that even if the alignment value of subsequences $R_1[0, ..., i_1]$, $R_2[0, ..., i_2]$,..., $R_N[0, ..., i_N]$ and the alignment value of subsequences $R_1[j_1, ..., n_1]$, $R_2[j_2, ..., n_2]$,...,

$R_N[j_N, ..., n_N]$ are maximal($\sum_{m=1}^{N}\sum_{l=1}^{N} d_{m,l}[i_m, i_l]$ and $\sum_{m=1}^{N}\sum_{l=1}^{N} d'_{m,l}[j_m, j_l]$), then the whole alignment value will be less than U:

$$D[\overrightarrow{i,j}] + \sum_{m=1}^{N}\sum_{l=1}^{N} d_{m,l}[i_m, i_l] + \sum_{m=1}^{N}\sum_{l=1}^{N} d'_{m,l}[j_m, j_l] < U \tag{10}$$

3 RNA Folding

The three-dimensional(secondary) structure of a RNA sequence has a functional role for many molecules and since knowing the sequence is not sufficient to determine the structure, it is important to be able to predict the most probable structures.

Definition 1 (Folding). *A secondary structure on a sequence $a = a_1, ..., a_n$ is a set of pairs (i, j), where $1 \leq i < j \leq n$ satisfying the knot constraint - if $i \leq i' \leq j \leq j'$, then (i, j) and (i', j') cannot be two distinct elements of S [5].*

3.1 Nussinov Algorithm

The Nussinov algorithm predicts the secondary structure of RNA by maximizing of the number of base pairs, according to the assumption that the more base pairs contains the sequence the more stable it is.

Let R be the input RNA sequence of lenth n. The algorithm fills in the matrix F of size $n * n$, where the value of $F[i, j]$ is the maximum number of possible base pairs in subsequence $R[i...j]$. The best structure for subsequence $R[i...j]$ is calculated from the previously calculated best structures for smaller subsequences [6]. There are four following ways to fill in the value of $F[i, j]$:

- Add unpaired base i to the best secondary structure for subsequence $R[i + 1, j](F[i, j] = F[i + 1, j])$.
- Add unpaired base j to the best secondary structure for subsequence $R[i, j - 1](F[i, j] = F[i, j - 1])$.
- Add base pair (i, j) to the best secondary structure for subsequence $R[i + 1, j - 1](F[i, j] = F[i + 1, j - 1] + \delta(i, j)$, where $\delta(i, j)$ is value of adding this kind of pair).
- Combination of the best secondary substructures for subsequences $R[i, k]$ and $R[k + 1, j]$ for some $k \in \{i + 1, ..., j - 1\}$. $(F[i, j] = F[i, k] + F[k + 1, j])$

This leads to following equations, where $1 \leq i \leq n$ and $i < j \leq n$:

$$F(i, j) = max \begin{cases} F[i + 1, j] \\ F[i, j - 1] \\ F[i + 1, j - 1] + \delta(i, j) \\ max_{i<k<j}\{F[i, k] + F[k + 1, j]\} \end{cases} \tag{11}$$

$$F(i, i) = F(i, i + 1) = 0 \tag{12}$$

The matrix is filled, beginning with the initialization according to (12) and then computing the lower half from smaller to longer subsequences [6]. After filling the matrix the solution is recieved via backtracking beginning with $F[1, n]$ what is the maximum number of base pairs for the whole sequence. The time complexity of Nussinov algorithm is $O(n^3)$ because of filling in n^2 cells by looking at $O(n)$ cells filled before. The space complexity is $O(n^2)$.

4 Automated Comparative Sequence Analysis

The secondary structure found by one of the algorithms from the previous section do not have to be the right one. We can predict the secondary structure much better if we know the alignment of given sequences. And we can find better multiple alignment if we know the consensus secondary structure. Usually we have only sequences and we know neither the alignment nor the secondary structure. If we want to find them both we can use three different approaches:

- Align sequences using classical multiple alignment tools and then predict the structure for found alignment according to structure neutral mutations. This method can be used only if sequences are well conserved. (RNAalifold [7][8], Pfold [9][10][11], ILM [12][13])
- If sequences are not well conserved, then predict seconadary structure for each sequence separately and directly align the structures(RNA forester [14][15], MARNA [16][17])
- Simultaneous alignment and folding(Sankoff [5])

The first two methods can lead to inaccurate results. This is why more complex algorithms were developed, that solves both problems in one time.

4.1 Sankoff Algorithm

Sankoff in 1985 introduced a dynamic programming algorithm [5] that combines the objective functions for alignment and minimal free energy folding to solve the alignment of finite sequences, their folding and the reconstruction of ancestral sequences on a phylogenetic tree.

Dynamic programming for complex problems is usually computationally expensive. Sankoff avoids this by the combination of biologically well motivated constraints on the set of possible alignment and folding solutions.

This algorithm is based on Sankoff's quadratic alignment algorithm ([18]), distance function for evaluating alignments by Sellers ([19] and [20]) and on the Zuker and Sankoff paper [21], who synthesize and advance a series of improvements in algorithmic efficiency including Nussinov algorithm and Zuker algorithm.

When biologists say that two RNA sequences have the same secondary structure, it does not mean that they are identical. For example the stems or loops does not have exactly the same length, but they can vary a bit. Let $i_1 < i_2 < ... < i_v$ be the positions in sequence $\mathbf{a}$ of all terms which form part either of

an external pair or of an accessible pair in a multiple loop in structure **A**. Let $j_1 < j_2 < ... < j_w$ be the same for structure **B** on sequence **b**. For **A** and **B** to be equivalent in terms of branching configurations, we require $v = w$ and $(i_f, i_g) \in A$ if and only if $(j_f, j_g) \in B$. Another requirement is that any k-loop have to be aligned with a single k-loop in the other structure, or be inserted or deleted in toto. [5]

The goal is to find equivalent structures and constrained alignment such that the entire found configuration is optimal. Sankoff introduced an objective optimization function representing a trade-off between free energy and alignment cost.

They extended the definition of $D(i_1, j_1; i_2, j_2)$(minimizing cost of an alignment between $a_{i_1}, ..., a_{j_1}$ and $b_{i_2}, ..., b_{j_2}$) to refer the cost of inserting the entire sequence $b_{i_2}, ..., b_{j_2}$ if $i_1 > j_1$ and to refer the cost of deleting $a_{i_1}, ..., a_{j_1}$ if $i_2 > j_2$.

Let $P(i_1, j_1; i_2, j_2)$ be the minimum cost for $(i_1, j_1) \in S_1$ and $(i_2, j_2) \in S_2$ without the cost of aligning $a_{i_1}^{(1)}$, $a_{j_1}^{(1)}$, $a_{i_2}^{(2)}$ and $a_{j_2}^{(2)}$. If no such pairs exists then $P(i_1, j_1; i_2, j_2) = \infty$.

$$P(i_1, j_1; i_2, j_2) = min \begin{cases} eH(i_1, j_1) + eH(i_2, j_2) + D(i_1 + 1, j_1 - 1; i_2 + 1, j_2 - 1)) \\ min_{i_1 < p_1 < q_1 < j_1; i_2 < p_2 < q_2 < j_2} \{eL(i_1, j_1, p_1, q_1) + eL(i_2, j_2, p_2, q_2) \\ \quad + P(p_1, q_1; p_2, q_2) \\ \quad + D(i_1 + 1, p_1; i_2 + 1, p_2) + D(q_1, j_1 - 1; q_2, j_2 - 1)\} \\ min_{i_1 + 1 < k_1 < j_1 - 1; i_2 + 1 < k_2 < j_2 - 1} \{M(i_1 + 1, k_1; i_2 + 1, k_2) \\ \quad + M(k_1 + 1, j_1 - 1; k_2 + 1, j_2 - 1) + 2 * A\} \end{cases} \tag{13}$$

$$M(i_1, j_1; i_2, j_2) = min \begin{cases} P(i_1, j_1; i_2, j_2) + 2 * B + D(i_1, i_1; i_2, i_2) + D(j_1, j_1; j_2, j_2) \\ min_{i_1 < k_1 < j_1; i_2 < k_2 < j_2} \begin{cases} (k_1 - i_1 + k_2 - i_2 + 1) * C \\ \quad + M(k_1 + 1, j_1; k_2 + 1, j_2) \\ M(i_1, k_1; i_2, k_2) \\ \quad + M(k_1 + 1, j_1; k_2 + 1, j_2) \\ M(i_1, k_1; i_2, k_2) \\ \quad + (j_1 - k_1 + j_2 - k_2) * C \end{cases} \end{cases} \tag{14}$$

Let $F(i_1, j_1; i_2, j_2)$ be the minimum cost possible for a pair of equivalent secondary structures S_1 and S_2 on positions $i_1, ..., j_1$ and $i_2, ..., j_2$ of sequences $a^{(1)}$ and $a^{(2)}$ respectively, where the cost is the sum of the free energies and a constrained alignment cost.

$$F(i_1, j_1; i_2, j_2) = min \begin{cases} P(i_1, j_1; i_2, j_2) + D(i_1, i_1; i_2, i_2) + D(j_1, j_1; j_2, j_2) \\ min_{i_1 \le k_1 < j_1; i_2 \le k_2 < j_2} \{F(i_1, k_1; i_2, k_2) + F(k_1 + 1, j_1; k_2 + 1, j_2)\} \\ D(i_1, j_1; i_2, j_2) \end{cases} \tag{15}$$

The initial conditions are $P(i_1, j_1; i_2, j_2) = \infty$ and $M(i_1, i_1; i_2, j_2) = M(i_1, j_1; i_2, i_2) = \infty$.

Lets take a look at the complexity of Sankoff algorithm for 2 sequences. The complexity of computing D is $O(n_1^2 * n_2^2)$ what is approximately $O(n^4)$ if $n_1 \approx$

$n_2 \approx n$. The time complexity of counting $P(i_1, j_1; i_2, j_2)$ is $O(n^2)$ if we bound possible loop length by some constatnt U. Time needed to count $M(i_1, j_1; i_2, j_2)$ and $F(i_1, j_1; i_2, j_2)$ is $O(n^2)$ too. Totally $O(n^4)$ cells of P, M, F matrices have to be filled. So the overal time complexity of the algorithm is $O(n^4 + 3 * n^6) = O(n^6)$ and space complexity is $O(4 * n^4) = O(n^4)$.

The equations (13), (14) and (15) can be generalized for N sequences as follows:

$$P(\overrightarrow{i,j}) = min \begin{cases} \sum_{r=0}^{N} eH(i_r, j_r) + D(\overrightarrow{i+1, j-1}) \\ min_{i_r < p_r < q_r < j_r} \{ \sum_{r=0}^{N} eL(i_r, j_r, p_r, q_r) + P(\overrightarrow{p,q}) \\ \quad + D(\overrightarrow{i+1, p}) + D(\overrightarrow{q, j-1}) \} \\ min_{i_r+1 < k_1 < j_r-1} \{ M(\overrightarrow{i+1, k}) \\ \quad + M(\overrightarrow{k+1, j-1}) + N * A \} \end{cases} \tag{16}$$

$$M(\overrightarrow{i,j}) = min \begin{cases} P(\overrightarrow{i,j}) + N * B + \gamma(a_{i_1}^{(1)}, a_{i_2}^{(2)}, ..., a_{i_N}^{(N)}) + \gamma(a_{j_1}^{(1)}, a_{j_2}^{(2)}, ..., a_{j_N}^{(N)}) \\ min_{i_r < k_r < j_r} min \begin{cases} \sum_{r=0}^{N} (k_r - i_r + 1) * C + M(\overrightarrow{k+1, j}) \\ M(\overrightarrow{i,k}) + M(\overrightarrow{k+1, j}) \\ M(\overrightarrow{i,k}) + \sum_{r=0}^{N} (j_r - k_r) * C \end{cases} \end{cases} \tag{17}$$

$$F(\overrightarrow{i,j}) = min \begin{cases} P(\overrightarrow{i,j}) + \gamma(a_{i_1}^{(1)}, a_{i_2}^{(2)}, ..., a_{i_N}^{(N)}) + \gamma(a_{j_1}^{(1)}, a_{j_2}^{(2)}, ..., a_{j_N}^{(N)}) \\ min_{i_r \leq k_r < j_r} \{ F(\overrightarrow{i,k}) + F(\overrightarrow{k+1, j}) \} \\ D(\overrightarrow{i,j}) \end{cases} \tag{18}$$

The initial conditions are $P(\overrightarrow{i,i}) = \infty$ and $M(\overrightarrow{i,j}) = \infty$ if any $i_r = j_r$. And the time complexity of generalized algorithm is $O(n^{3*N})$ and it needs $O(n^{2*N})$ space.

4.2 Simultaneous Alignment and Nussinov Folding

Our objective is to explore, whether it is possible to apply the Carrillo-Lipman approach for multiple sequence alignment to speed up simultaneous alignment and folding. We decided to try it at first with the Nussinov folding algorithm. The first step to this is to design an equation for simultaneous alignment and Nussinov folding.

We combine the Sankoff quadratic alignment algorithm with Nussinov folding algorithm like Sankoff[5] combined it with Zuker folding algorithm. For two sequences we distinguish five cases when filling in the value of $F[i, j; k, l]$:

- Add pairs of bases i, j and k, l, base i will be aligned to base k and base j will be aligned to base l. Then the value will be computed as sum of the value $F[i + 1, j - 1; k + 1, l - 1]$, the values of created base pairs $\delta(i, j) + \delta(k, l)$, and the value of aligned bases $C(R_{1,i}, R_{2,k}) + C(R_{1,j}, R_{2,l})$
- Align unpaired nucleotides i and k to the best aligned structures of sequences $R_1[i + 1, ..., j]$ and $R_2[k + 1, ..., l]$.

- Align unpaired nucleotides j and l to the best aligned structures of sequences $R_1[i, ..., j-1]$ and $R_2[k, ..., l-1]$.
- There are no base pairs in subsequences $R_1[i, ..., j]$ and $R_2[k, ..., l]$ so we have just to count the value of aligning those sequences.
- Combine two previously counted substructures $F[i, h; k, p]$ and $F[h+1, j; p+1, l]$.

This four cases leads to the following equation:

$$F[i,j;k,l] = max \begin{cases} F[i+1, j-1; k+1, l-1] + \delta(i,j) + \delta(k,l) \\ \quad +C(R_{1,i}, R_{2,k}) + C(R_{1,j}, R_{2,l}) \\ F[i+1, j; k+1, l] + C(R_{1,i}, R_{2,k}) \\ F[i, j-1; k, l-1] + C(R_{1,j}, R_{2,l}) \\ D[i,j;k,l] \\ max_{i<h<j;k<p<l} F[i,h;k,p] + F[h+1, j; p+1, l] \end{cases} \tag{19}$$

$$\forall i,j : F[i, i-1; j, j-1] = 0 \tag{20}$$

The space complexity here is $O(n^4)$ and the time complexity goes to $O(n^6)$ because of the fifth case.

We can also generalize it for N sequences (21). It needs $O(n^{2*N})$ space and $O(n^{3*N})$ time.

$$F[\overrightarrow{i,j}] = max \begin{cases} F[\overrightarrow{i+1, j-1}] \\ \quad + \sum_{k=1}^{N} \delta(i_k, j_k) \\ \quad + \sum_{m=1}^{N} \sum_{k=m+1}^{N} (C(R_{m,i_m}, R_{k,i_k}) + C(R_{m,j_m}, R_{k,j_k})) \\ F[\overrightarrow{i+1, j}] + \sum_{m=1}^{N} \sum_{k=m+1}^{N} C(R_{m,i_m}, R_{k,i_k}) \\ F[\overrightarrow{i, j-1}] + \sum_{m=1}^{N} \sum_{k=m+1}^{N} C(R_{m,j_m}, R_{k,j_k}) \\ D[\overrightarrow{i,j}] \\ max_{i_m<h_m<j_m} F[\overrightarrow{i,h}] + F[\overrightarrow{h+1, j}] \end{cases} \tag{21}$$

4.3 Applying Carrillo-Lipman Approach to Simultaneous Alignment and Nussinov Folding

Suppose we have some unoptimal alignment and folding of sequences $R_1, R_2, ..., R_n$ with value U. We want to apply the Carrillo-Lipman approach to our algorithm. At first we have to find the upper bound for the value of simultaneous alignment and folding. The upper bound of alignment value for subsequences $R_1[0, ..., i_1]$, $R_2[0, ..., i_2]$, ... , $R_N[0, ..., i_N]$ is the sum of maximal pairwise alignments values for all pairs of sequences, same like in Carrillo-Lipman algorithm. The upper bound for simultaneous algorithm will be equal to sum of the upper bound for alignment and the upper bound for secondary structures. The value of secondary structure is added only in the first case. The value of every base pair in every sequence is added exactly once. Let $f_m(i_m, j_m)$ be the value of best secondary structure of subsequence $R_m[0, ..., i_m - 1, j_m + 1, ..., n_m]$. So we can say, that:

$$F[\overrightarrow{0,n}] - \sum_{m=1}^{N} f_m(i_m, j_m)) - \sum_{m=1}^{N} \sum_{l=1}^{N} d_{m,l}[i_m, i_l] - \sum_{m=1}^{N} \sum_{l=1}^{N} d'_{m,l}[j_m, j_l] \leq F[\overrightarrow{i,j}] \tag{22}$$

The cell $F[\overrightarrow{i,j}]$ can then be excluded from the computation if its value is so small that even if the alignment and folding value of subsequences $R_1[0,...,i_1,j_1,...,n_1]$, $R_2[0,...,i_2,j_2,...,n_2]$, ... , $R_N[0,...,i_N,j_N,...,n_N]$ are maximal:

$$\sum_{m=1}^{N}\sum_{l=m+1}^{N} d_{m,l}[i_m,i_l] + \sum_{m=1}^{N} f_m(i_m,j_m) + \sum_{m=1}^{N}\sum_{l=m+1}^{N} d'_{m,l}[j_m,j_l] \tag{23}$$

then the whole alignment and folding value will be less than U:

$$F[\overrightarrow{i,j}] + \sum_{m=1}^{N} f_m(i_m,j_m) + \sum_{m=1}^{N}\sum_{l=m+1}^{N} d_{m,l}[i_m,i_l] + \sum_{m=1}^{N}\sum_{l=m+1}^{N} d'_{m,l}[j_m,j_l] < U \tag{24}$$

The preprocessing complexity depends on the heuristical method used for finding the unoptimal alignment and folding. Except of that it needs $O(2*N^2*n^2) = O(N^2*n^2)$ space and $O(2*N^2*n^2) = O(N^2*n^2)$ time for pairwise alignment using Needleman-Wunsch algorithm, and $O(N*n^2*n^2) = O(N*n^4)$ space and $O(N*n^2*n^3) = O(N*n^5)$ time for finding best secondary structure for each sequence using Nussinov algorithm. The most expensive part is counting the Sankoff multiple alignment matrix D. It needs $O(n^{2*N})$ space and $O(n^{2*N} * 2^N * N^2)$ time, but it is also a part of the computation of unimproved algorithm.

This approach allows us to simultaneously align and fold sequences in more reasonable time. Testing results for 3 sequences from RNA family HIV-1_SD are in following table. We can see, that increasing the value of U rapidly decreased number of counted cells in F. First used value of U (49.0) was the value of multiple alignment($D[\overrightarrow{0,n}]$) counted in preprocessing phase. The value of simultaneous alignment and folding is never less than $D[\overrightarrow{0,n}]$.

Table 1. Computation times for HIV-1_SD family

U	# counted cells	# skipped cells	preprocess.	main comput.	totall time	result
49.0	457 062	9 600 830	87 s	1 687 s	1 774 s	71.0
55.0	143 061	9 196 462	86 s	217 s	303 s	71.0
60.0	36 410	9 046 724	85 s	145 s	230 s	71.0
65.0	3 889	9 002 811	85 s	140 s	225 s	71.0

5 Conclusion

In this work we proposed an algorithm for simultaneous alignment and folding using the Nussinov folding algorithm. We also proposed a speed-up of this algorithm that cuts off unnecessary computations according to Carrillo-Lipman multiple sequence alignment algorithm. Our approach leads to an optimal solution of alignment and folding for more than two sequences using the Nussinov scoring scheme for folding in better time than the Sankoff simultaneous alignment and folding algorithm. In our future work we want to find out whether it is possible to use this approach with more complex Zuker folding scheme [21].

References

1. Edgar, R.C., Batzoglou, S.: Multiple sequence alignment. Current Opinion in Structural Biology 16, 368–373 (2006)
2. Wang, L., Jiang, T.: On the complexity of multiple sequence alignment. J. Comput. Biol. 1, 337–348 (1994)
3. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 48, 443–453 (1970)
4. Carrillo, H., Lipman, D.: The multiple sequence alignment problem in biology. SIAM Journal on Applied Mathematics 48 (1988)
5. Sankoff, D.: Simultaneous solution of the RNA folding, alignment, and protosequence problems. SIAM J. Appl. Math. 45, 810–825 (1985)
6. Sperschneider, V., Sperschneider, J., Scheubert, L.: Bioinformatics: problem solving paradigms. Springer (2008)
7. Hofacker, I., Fekete, M., Stadler, P.: Secondary structure prediction for aligned RNA sequences. Journal of Molecular Biology 319(5), 1059–1066 (2002)
8. RNAalifold: `http://www.tbi.univie.ac.at/ivo/RNA/`
9. Knudsen, B., Hein, J.: Pfold: RNA secondary structure prediction using stochastic context-free grammars. Nucleic Acids Research 31(13), 3423–3428 (2003)
10. Knudsen, B., Hein, J.: RNA secondary structure prediction using stochastic context-free grammars and evolutionary history. Bioinformatics 15(6), 446–454 (1999)
11. Pfold, `http://www.daimi.au.dk/compbio/rnafold/`
12. Ruan, J., Stormo, G., Zhang, W.: An iterated loop matching approach to the prediction of RNA secondary structures with pseudoknots. Bioinformatics 20, 58–66 (2004)
13. ILM, `http://www.cs.wustl.edu/zhang/projects/rna/ilm/`
14. Höchsmann, M., Töller, T., Giegerich, R., Kurtz, S.: Local similarity of RNA secondary structures. In: Proc of the IEEE Bioinformatics Conference, pp. 159–168 (2003)
15. RNAforester, `http://bibiserv.techfak.uni-bielefeld.de/rnaforester/`
16. Siebert, S., Backofen, R.: MARNA A server for multiple alignment of RNAs. In: Proceedings of the German Conference on Bioinformatics, pp. 135–140 (2003)
17. MARNA, `http://www.bio.inf.uni-jena.de/Software/MARNA/index.html`
18. Sankoff, D.: Matching sequences under deletion/insertion constraints. Proceedings of the National Academy of Sciences (U.S.A.) 69, 4–6 (1972)
19. Sellers, P.H.: An algorithm for the distance between two finite sequences. J. Combin. Theory 16, 252–258 (1974)
20. Sellers, P.H.: On the theory and computation of evolutionary distances. SIAM J. Appl. Math. 26, 787–793 (1974)
21. Zuker, M., Sankoff, D.: RNA secondary structure and their prediction. Bull. Math. Biol. 46, 591–621 (1984)

Large Scale Metabolic Characterization Using Flux Balance Analysis and Data Mining

Miguel Rocha

CCTC/ Dep. Informatics
Campus Gualtar, Universidade do Minho, Braga, Portugal
mrocha@di.uminho.pt

Abstract. Genome-scale metabolic models of several microbes have been reconstructed from sequenced genomes in the last years. These have been used in several applications in Biotechnology and biological discovery, since they allow to predict the phenotype of the microorganism in distinct environmental or genetic conditions, using for instance Flux Balance Analysis (FBA). This work proposes an analysis workflow using a combination of FBA and Data Mining (DM) classification methods, aiming to characterize the metabolic behaviour of microorganisms using the available models. This framework allows the large scale comparison of the metabolism of different organisms and the prediction of gene expression patterns. Also, it can provide insights about transcriptional regulatory events leading to the predicted metabolic behaviour. DM techniques, namely decision tree and classification rules inference, are used to provide patterns of gene expression based on environmental conditions (presence/ absence of substrates in the media). The methods proposed are applied to the study of the metabolism of two related microbes: *Escherichia coli* and *Salmonella typhimurium*.

Keywords: Genome-scale Metabolic Models, Transcriptional Regulation, Systems Biology, Flux Balance Analysis, Rule inference.

1 Introduction

Recent efforts in Bioinformatics and Systems Biology allowed the development of genome-scale metabolic models for several microorganisms [4]. These models have been used to guide biological discovery promoting the comparison between predicted and experimental data, to foster Metabolic Engineering efforts in finding appropriate genetic modifications to synthesize desired compounds, to analyze global network properties and to study bacterial evolution [5].

The most popular approach to phenotype simulation considers the cell to be in steady-state and takes reaction stoichiometry/ reversibility in a constraint-based framework to restrict the set of possible values for the reaction fluxes. Cellular behaviour is thus predicted using for instance Flux Balance Analysis (FBA), based on the premise that microorganisms have maximized their growth along natural evolution, a premise that has been validated experimentally [6].

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 336–345, 2013.

Using FBA, it is possible to predict the behaviour of microbes under distinct environmental/ genetic conditions.

The aim of this paper is to use metabolic models and phenotype simulation methods to provide for large scale analysis of the metabolism of an organism, to allow the prediction of gene expression patterns consistent with those data and, finally, to gain insights on the transcriptional regulatory events that need to be in place to achieve the determined metabolic behaviours. These tools can be applied to several organisms enabling the comparative analysis of their metabolic and genetic behaviours. Data Mining (DM) tools will play a central role in extracting patterns from the large datasets of simulations provided by FBA.

Regarding regulation, the basic idea of this paper is based on the following question: if the metabolic behaviour of an organism can be determined, what can we infer about its regulatory machinery? In other words, what kind of regulatory mechanisms do we need to provide in order to explain a given metabolic behaviour. We will assume that FBA provides a reasonable approximation to the metabolic behaviour and will try to infer some features about the regulatory machinery that could lead to this behaviour.

In the experiments provided in this work, we will use two microbes, both with validated genome-scale metabolic models: *Escherichia coli* and *Salmonella typhimurium*. The first is a well known bacterium that also has a well established regulatory model that will serve as a gold standard; the second is an important pathogen, with high similarity to *E. coli* but with no regulatory model available.

Given the two organisms and their genome-scale metabolic models, the simulation of the phenotype in a wide range of environmental conditions is performed. The set of environmental conditions is created to span a large number of possible media where the organisms can grow. For each condition, a simulation is performed using FBA, allowing the determination of which reactions and genes will be active under those conditions. This allows the creation of a map, indicating the pattern of expression expected for each metabolic gene and, consequently, the determination of the overall activity of each gene, i.e. in what proportion of conditions it is active. Since one of the focus of the study will be on the genes that are a target of regulation, all genes that are expressed in nearly all conditions or almost none are filtered from further analyses. Restricting to the remaining genes, the next logical step is to determine groups of genes that show a similar pattern of co-expression, i.e. are expressed in the same set of conditions.

Also, taking into account, for each condition, which is the set of substrates (external metabolites) that are available for uptake by the organism, patterns of gene expression based on the external metabolites presence/ absence can be extracted. This is achieved by the use of Data Mining (DM) classification algorithms, taking as inputs a set of binary attributes representing the presence/ absence of a given substrate and as outputs the expression/ not expression of a given gene. These classifiers are given by decision rules with a condition including the presence/ absence of substrates on the left hand side and the binary expression value of the gene in the right hand side. This allows checking the dependencies of gene expression on the environmental conditions.

If the target organism has an available regulatory model or a set of known regulatory interactions, these results can be compared to this information, by:

- Comparing clusters of co-expressed genes from the simulations, with sets of genes sharing regulatory mechanisms (e.g. controlled by the same transcription factors);
- Comparing the dependencies of the gene expression on external metabolites with the reciprocal variables that constrain the regulatory model for the same genes.

2 Methods and Models

2.1 Models

In this work, the following genome-scale models were used:

- a metabolic model for *Escherichia coli* - *iAF1260*, containing 1260 genes, 2077 reactions and 1039 metabolites [3];
- a metabolic model for *Salmonella typhimurium* - *iMA945*, including 945 genes, 1964 reactions and 1036 metabolites [1];
- regulatory model for *E. coli* - *iMC1010* - proposed by Covert et al [2], containing 1010 genes, integrated with the corresponding metabolic model.

The metabolic models contain information about all the reactions that can occur in the cell, including the substrates and products, their stoichiometry and reversibility. Also, for each reaction the corresponding gene-reaction rule is given, which is a Boolean expression stating if the reaction is active as a function of the genes involved in the encoding of the respective enzymes.

The regulatory model is also given in terms of Boolean logic: the left hand side corresponds to regulated genes, while the right-hand sides are Boolean expressions involving regulatory genes and external variables (e.g. presence/ absence of metabolites in the media or other events such as stress responses).

To allow the comparison between the two organisms, a table of orthologous genes was created, enabling to convert gene mentions from one organism to the other. This table was generated using the well known Bidirectional Best Hit method, that works as follows for every gene :

1. for each gene in organism 1 (noted A), perform an homology search (using for instance BLAST) against the genome of organism 2 and keep the best hit (B);
2. B is then used as a query and a search against the genome of organism 1 is done achieving the best hit A*;
3. if A* is the original gene A, then A and B are orthologs.

Using this method, it was possible to find 871 ortholog pairs (93% of the total number of genes in the *S. typhimurium* model).

2.2 Flux Balance Analysis

The Flux Balance Analysis (FBA) [6] approach is based on a steady state approximation to the concentrations of internal metabolites, which reduces the corresponding mass balances to a set of linear homogeneous equations. For a network of M metabolites and N reactions, this is expressed as:

$$\sum_{j=1}^{N} S_{ij} v_j = 0 \tag{1}$$

where S_{ij} is the stoichiometric coefficient for metabolite i in reaction j and v_j the flux over the reaction j. The limits of the fluxes can be set by additional constraints in the form $\alpha_j \leq v_j \leq \beta_j$, also used to specify nutrient availability.

The set of linear equations obtained usually leads to an under-determined system. However, if a linear function over the fluxes is chosen to be maximized, it is possible to obtain a single solution by applying standard algorithms (e.g. *simplex*) for Linear Programming. The most common flux chosen for maximization is the biomass given the premise of optimal evolution that underlies FBA.

2.3 Workflow

The workflow for the *in silico* analysis conducted in this work was the following:

1. A large set of environmental conditions (ECs) were created (corresponding to different media). A set of external metabolites (carbon/ nitrogen sources) was identified (list given on supplementary material). ECs are created as all combinations of a carbon and a nitrogen source. Aerobic and anaerobic variants of each EC were created. Some metabolites are always considered to be present, namely SO_4, Pi, H_2O, H, $ions$, CO_2.
2. Using FBA, phenotype simulations were conducted using each of the metabolic models (*E. coli* and *S. typhimurium*) for those ECs. The flux value for each reaction in each of the simulations was determined.
3. All simulations where the organism does not grow (biomass flux less than 1% of the typical wild type growth in glucose) were removed.
4. The simulation results were binarized, i.e. changed to value 1, if reaction has flux larger than zero in that simulation and 0 otherwise. This step creates a binary matrix of ECs $\times$ reactions
5. Using the gene-reaction rules from the metabolic models, a set of entailments was defined in the form $IF(reaction = On) \Rightarrow (gene = On)$. Only cases where this can be assumed with 100% certainty are used: cases of one gene-one reaction; cases where reaction is On and rule is a logical conjunction.
6. Based on these entailments, and taking as input the binary matrix of reactions, the value of each metabolic gene in the model, for each condition was determined. The possible alternatives are:
   - ON - if the state of the gene is determined by one of the gene entailments
   - OFF - if no gene entailment can provide a value for the gene

   The result is a matrix of ECs x genes, where each cell has value ON or OFF.

7. Number of ECs where each gene is ON was calculated; genes were ranked.
8. The expression levels of genes (from the last step) in *E. coli* were compared with the levels of orthologous genes in *S. typhimurium*.
9. The binary gene matrix was filtered, removing genes that are very frequently ON (more than 99% of the ECs) and genes that are very rarely ON (less than 1%). The remaining genes are the ones that change significantly when the environment changes; these were chosen to study how their expression is regulated. Orthologous genes that are present in both lists were identified.
10. For all pairs of the remaining genes, co-expression values were computed, using Jaccard coefficient (JC). For genes (g_1 and g_2), JC is given by:

$$JC(g_1, g_2) = \frac{M_{11}}{M_{11} + M_{10} + M_{01}} \tag{2}$$

where: M_{11} is the number of conditions where both genes are ON; M_{10} is the number of conditions where g_1 is ON and g_2 is OFF; M_{01} is the number of conditions where $g_1$is OFF and g_2 is ON.

11. All pairs of genes with JC larger than a given threshold were listed. A graph was defined with all those genes, where each pair is connected by a link if the JC is larger than the threshold (in this case, 97%). Each connected component of this graph is identified to provide a cluster. Resulting clusters were analysed computing:
    (a) the mean value of the percentage of ECs for which the genes in the cluster are expressed (measures the level of expression of the cluster);
    (b) the mean Jaccard value for all pairs of genes in the cluster (this measures the compactness of the cluster);
    (c) the percentage of all possible pairs of genes in the cluster, where the Jaccard value is larger than the threshold (measures the coherence).
12. The clusters obtained using the *E. coli* and the *S. typhimurium* model were compared, following the steps below:
    (a) for each *S. typhimurium* cluster, we calculated an equivalent cluster transforming all genes in their orthologs in *E. coli*;
    (b) for each of those clusters we searched for the cluster in *E. coli* that has the largest intersection (if any);
    (c) for each of these matches, we calculated the size of the intersection, the coverage of the intersection in both sets and the JC;
    (d) global statistics for all clusters were generated.
13. The results of step 11 were compared with the regulatory model for *E. coli*:
    (a) checking pairs of genes that share the same rule in the regulatory model and comparing these pairs with the ones from step 11;
    (b) creating clusters from the pairs in the regulatory model in a way similar to the one followed in step 11;
    (c) comparing clusters of *E. coli* from step 11 with clusters in the regulatory model;
    (d) comparing clusters of *S. typhimurium* with regulatory clusters in *E. coli*.

14. For each gene, Data Mining algorithms were used to infer rules in the form: gene is *ON* if external metabolite is present/absent in the EC. The idea is to characterize all the conditions for which a given gene is expressed. Two classifiers taken from the Weka DM platform [7] are used: (i) J48, which provides a decision tree; (ii) JRip, which provides a set of rules. The latter was selected since it provided more accurate classifiers, in preliminary tests using 10-fold cross validation (results not shown). This process was done in two steps:
    (a) the rules for the genes in a given cluster (from step 11) will be very similar and therefore only one set of rules is extracted for each cluster;
    (b) rules were extracted for each gene not belonging to any clusters.
15. The DM classifiers obtained for both organisms were compared. The comparison is based on the set of external metabolites in the right hand side of the rules from the previous step. The list of genes to be considered will be the intersection of the genes in both models (from step 9).
16. For each cluster where there was a match (complete or partial) in step 13 c) or d), we compared the set of external metabolites in the rules extracted in step 14 with the set of external metabolites that control the gene expression in the matching cluster in the model (including all upstream transcription factors). The process was repeated for the list of unclustered genes (step 14) and all genes in clusters where there was no match with the model.

3 Results

In this section, an overview of the main results obtained by following the previous workflow is provided. The supplementary files mentioned along the text and the full results of the study are provided in the URL: `http://darwin.di.uminho.pt/icannga13`. All results files for a single organism are given in the form *model-filename.txt*, where *model* is *EC* for the results related to the *E. coli* model and it is *ST* for the results obtained using the *S. typhimurium* model.

3.1 Environmental Conditions and Growth Analysis

In the experiments, to create the ECs, 71 carbon sources and 38 nitrogen sources (files *carbonSources.txt* and *nitrogenSources.txt*) were identified and used in both models, providing a total of 5354 distinct ECs (step 1). These were organized in a binary substrate matrix of ECs $\times$ external metabolites, indicating which external metabolites (substrates) are taken from the media in a given EC (files named *model-substrateMatrix.txt* in supplementary material).

The results of step 2 indicate the growth values obtained running FBA in the set of ECs for each of the models. The file *ECST-comparisonBiomass.xlsx* summarizes these results, showing the growth value obtained in each EC for each of the models. From the simulations, 3885 ECs resulted in growth for *E. coli*, while in the case of *S. typhimurium* the number is 3462 (73% and 65% respectively). These are the ECs kept for further analysis (step 3).

Also, we can calculate the overlap of the conditions where both models have a similar outcome (growth/ no growth) to be around 82% showing a high level of similarity between the metabolic behaviour of both organisms. To further analyse these results, we used DM classifiers to characterize the conditions where one of the organisms is able to grow, while the other is unable to do it. The idea is to have a set of rules that represent which ECs are in those conditions, thus highlighting the major differences in the overall metabolic capabilities of both organisms. The results from the classifier are shown in file *ECST-comparisonGrowth.txt*. Step 4 creates a binary matrix of ECs × reactions showing which reactions are active in each simulation (files *model-fluxesBinary.txt*).

3.2 Gene Expression Analysis

Steps 5 and 6 provide results similar to the previous at the level of the genes, creating binary matrices of ECs × genes (files *model-GenesMatrix.txt*). Based on these data, the activity level of each gene is calculated, i.e. the percentage of ECs where the gene is expressed (files *model-GeneActivity.txt* (step 7). The genes were divided into classes according to their frequency. Results are given in Table 1. After filtering the very rarely and very frequently expressed genes, the number of genes kept for the analysis was 290 for *E. coli* and 256 for *S. typhimurium* (23% and 27% of the genes in the model, respectively). One conclusion is that the number of genes that are regulated seems to be relatively low. The main part of the genes are either always active or inactive.

Table 1. Analysis of gene expression level for both organisms

Level	*S. typhimurium*	*E. coli*
Never expressed	457 (49%)	748 (59%)
Rare (<5%)	144 (15%)	148 (12%)
Low (5-50%)	68 (7%)	100 (8%)
High (50-95%)	54 (6%)	59 (5%)
Very High (95%-99.9%)	49 (5%)	44 (3%)
Always (100%)	164 (18%)	162 (13%)

To further analyse these data, we checked the expression of the orthologous genes from the two organisms. The expression levels of each pair of orthologs was compared. The results are given in file *ECST-OrthologsExpression.xlsx* A good match between the two models is clear, measured by the low differences in the overall expression and the high values of the intersection of the expression patterns (measured by the application of the Jaccard coefficient to each pair). From this analysis it was possible to identify two groups: a large group with genes that are highly co-expressed in both organisms (high JC and low differences of global expression levels) and a small group that shows significant differences between the expression patterns in both organisms.

3.3 Co-expression Analysis and Cluster Formation

The Jaccard coefficient was used to identify pairs of co-expressed genes (step 10) that were further grouped into clusters of genes with high JC values (step 11). This analysis allowed to reach the number of clusters given in Table 2 for each organism. An analysis of some metrics related with the clusters cohesiveness and compactness (calculated in step 11) are summarized in files *model-ClusterAnalysis.txt*. Summarizing the contents of these files, the clusters found are quite compact (most of the possible intra-cluster interactions are present) and the mean Jaccard value of the possible pairs is quite elevated.

Table 2. Number of clusters found for each model

	S. typhimurium	*E. coli*
Number of clusters	48	59
Genes within clusters	149 (58%)	190 (66%)

The comparison of the clusters for the two organisms (considering the orthologous pairs) shows a high level of agreement. Indeed, when considering the match of the orthologs, 81% of the clusters have a partial match, while 52% have a full match. Also, the coverages of cluster elements from one organism to the other are 77% and 74%. The full metrics are given in *ECST-OrthologsClustersMatch.txt*.

3.4 Comparison with the Regulatory Model

The clusters previously determined for *E. coli* are analysed by comparing with the known transcriptional regulatory network (TRN) for the same organism (step 13). To enable this comparison, clusters are created from the TRN by gathering genes that are co-regulated by the same transcription factors (i.e. share the same right hand side of the regulatory rule in the TRN).

A comparison of the clusters from the TRN with the ones reached in step 11 leads to the conclusion that our method is able to recover a significant number of regulatory interactions. In fact, 59% of the clusters have a partial match and 25% have a complete match, while the coverage of the TRN clusters by our clusters is around 49%. The full analysis is provided in file *EC-MatchClustersTRN.txt*. The same exercise was provided for *S. typhimurium* (step 13 d). While no TRN is available, the one for *E. coli* is used as a template (assuming that it will be similar). The results are somewhat surprising, since the match of the clusters for *S. typhimurium* with the *E. coli* TRN is even better than the *E. coli* itself. Indeed, 65% of the clusters have a partial match and 27% have a complete match, while the coverage is around 55%. The full analysis results are given in *ST-MatchClustersTRN.txt*. This provides a strong evidence of the possible conservation of the TRN in both organisms, a fact that can be used to help in the reconstruction of a TRN for *S. typhimurium*.

3.5 Data Mining Analysis of Gene Expression

In step 14, DM classifiers are used to get rules for each gene, in terms of the substrates used in the ECs. This allows to infer which genes are necessary for growth in a given condition. The analyses provided for each cluster are given in files *model-ClusterDMAnalysis.txt* for the *JRip* classifier. Also, the same results were also calculated for individual genes not present in any cluster, being given in files *model-NoClusterDMAnalysis.txt*. A comparison was done of the classifiers obtained for both organisms, taking into account the set of attributes (external metabolites) used in each pair of orthologous genes. The result is given in file *ECST-DMAttributesMatch.txt*. An analysis of these results shows again a high level of agreement: 90% of partial match and 47% of full match.

Finally, we have compared the attributes used in the rules extracted using DM (for each cluster) with the available regulatory information. To achieve this, the rules available in the *E. coli* TRN were analysed and all dependencies of regulated gene to external metabolite presence/ absence were collected. Thus, for each gene, a set of external metabolites that influence its expression pattern was gathered. This was compared with the set of attributes used in the DM classifier of the same gene. The process was repeated for *E. coli* and *S. typhimurium*. Results are provided in the files: *model-DMAttsMatchTRN.txt*.

The results are summarized in Table 3 for both models. The high values obtained show that the proposed methods are able to correctly identify a significant number of regulatory interactions between the presence/absence of certain metabolites in the media and gene expression.

Table 3. Results of the comparison of DM attributes with TRN dependencies on external metabolites

	S. typhimurium	*E. coli*
Partial match	81%	83%
Full match	14%	13%
Coverage DM clusters	61%	55%
Coverage TRN clusters	42%	525

4 Conclusions and Further Work

In this work, a workflow for the large scale analysis of the metabolic behaviour of organisms where a genome-scale model exists has been proposed. These methods allow to characterize the gene expression of metabolic genes, based on the compounds available in the environment. We have shown that this method can be used to infer regulatory interactions, in the form of clusters of co-expressed genes, and also to infer the control of gene expression by the presence/ absence of external metabolites in the media. In the case studies, the methods allowed to recover a significant number of regulatory interactions that were confirmed

by an existing regulatory network. Also, the use of DM classifiers for external metabolite dependency identification is a powerful technique to characterize the space of media where a given gene/ reaction is required. Overall, these methods seem a good way to establish hypotheses on regulatory events.

This work suggests an approach for recovering regulatory interactions in *S. typhimurium* from the template TRN of *E. coli* that would involve the steps:

- check orthologous genes that show the same pattern of expression;
- check the regulatory interactions known for those genes in *E. coli*;
- infer that the same type of interactions exist in *S. typhimurium*;
- validate with expression data or pre-existent knowledge in databases.

This method can be very useful in uncovering regulatory models, a Bioinformatics task far less studied comparing to the reconstruction of metabolic models. Also, the results obtained in these large scale simulations can be used to validate the metabolic model by comparing patterns from model predictions to the measurements of gene expression (e.g. from DNA microarrays or RNAseq).

In future work, we intend to enlarge the methods proposed to a broader set of organisms, providing tools for the use of genome-scale metabolic models in comparative analysis of metabolic behaviour and regulatory mechanisms.

Acknowledgements. The work is partially funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT (Portuguese Foundation for Science and Technology) within projects ref. COMPETE FCOMP-01-0124-FEDER-015079 and PEst-OE/EEI/UI0752/2011.

References

1. AbuOun, M., Suthers, P.F., Jones, G.I., Carter, B.R., Saunders, M.P., Maranas, C.D., Woodward, M.J., Anjum, M.F.: J. Biol. Chem. 284(43), 29480–29488 (2009)
2. Covert, M.W., Knight, E.M., Reed, J.L., Herrgard, M.J., Palsson, B.O.: Integrating high-throughput and computational data elucidates bacterial networks. Nature 429(6987), 92–96 (2004)
3. Feist, A.M., Henry, C.S., Reed, J.L., Krummenacker, M., Joyce, A.R., Karp, P.D., Broadbelt, L.J., Hatzimanikatis, V., Palsson, B.O.: A genome-scale metabolic reconstruction for escherichia coli k-12 mg1655 that accounts for 1260 orfs and thermodynamic information. Molecular Systems Biology 3, 121 (2007)
4. Feist, A.M., Herrgard, M.J., Thiele, I., Reed, J.L., Palsson, B.O.: Reconstruction of biochemical networks in microorganisms. Nature Rev Microbiology 7(2), 129 (2008)
5. Feist, A.M., Palsson, B.O.: Nature Biotechnology 26(6), 659–667 (2008)
6. Ibarra, R.U., Edwards, J.G., Palsson, B.G.: Escherichia coli k-12 undergoes adaptive evolution to achieve in silico predicted optimal growth. Nature 420, 186–189 (2002)
7. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kauffman (2005)

Automatic Procedures to Assist in Manual Review of Marine Species Distribution Maps

Gianpaolo Coro[1], Pasquale Pagano[1], and Anton Ellenbroek[2]

[1] Istituto di Scienza e Tecnologie dell'Informazione "Alessandro Faedo" – CNR Pisa, Italy
{gianpaolo.coro,pasquale.pagano}@isti.cnr.it
[2] FAO - Food and Agriculture Organization
anton.ellenbroek@fao.org

Abstract. Ecological Niche Modeling (ENM) is a branch of biology that uses algorithms to predict the distribution of species in a geographic area on the basis of a numerical representation of their preferred habitat and environment. Algorithmic maps can be produced for suitable or native habitats and require a review by human experts. During the review operation biologists use their knowledge about a species to modify the maps. They usually take algorithmic maps as starting point in the review. In this paper we provide a methodology for biologists to use the automatic maps as references also during and after the review process. Our approach is based on a comparison between the reviewed map and two systems: an expert system and a Feed Forward Neural Network. Furthermore we suggest an evaluation procedure of the quality of the environmental features used as training set, for assessing the models reliability.

Keywords: Ecological Niche Modeling, Feed Forward Neural Networks, AquaMaps, Maps Review.

1 Introduction

The term Ecological Niche Modeling (ENM) refers to a set of methods that use algorithms to predict the distribution of species in a geographic area. ENM techniques focus on a numerical representation of a species habitat, which should be complete and made up of independent variables. Several approaches have been used for this task, ranging from physiological to mathematical models [16]. Results are usually produced in the form of GIS heat maps, where a color gradient represents a range of probabilities and hotter colors correspond with higher probabilities. These maps can be generated by automatic procedures using different kinds of approaches [15]. Automatically generated maps are usually reviewed by a biologist in order to correct inconsistencies. The biologist uses his/her knowledge about the species to make little corrections or to completely redesign the distribution map. Such corrections can consist of adjusting models parameters or in manually editing distribution values.

In this article, we propose a methodology that can help biologists in discovering gaps in their knowledge when reviewing a map. They can investigate

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 346–355, 2013.

inconsistencies and revise some assumptions made during a review. Furthermore, the method proposes statistical analysis techniques for validating the quality of the map. The method has been applied to marine species. It is based on a comparison between the reviewed map and two systems: an automatically generated distribution based on Feed Forward Neural Networks, and an expert system (Aquamaps [12]) for marine species distribution prediction. A features processing technique, based on Principal Component Analysis, is then used for assessing the reliability of the models and consequently of the reviewed map.

In Section 2 we report an overview of common approaches for producing species maps. In Section 3 we describe the methodology and propose an example of application. In Section 4 we draw the conclusions.

2 Overview

Ecological Niche Modeling is usually a complex and iterative process including [8]: (*i*) identification of *relevant data*, (*ii*) *modeling*, (*iii*) *projection* of predictions onto a geographic space. The first step is crucial and usually involves the identification of environmental features related to species preferences. In this paper we don't focus on features selection as our method is applied after the projection step. For what regards modeling, techniques are usually based on occurrence records (*presence points*), i.e. places where the species has been observed in its habitat. Some approaches need even to use *absence points*, i.e. locations where the environment is considered unsuitable for a species [9]. Models need representative occurrence data and independent and complete environmental parameters. These possibly give robustness and reliability to the models [11,8]. The choice of a suitable modeling technique for a specific scenario is not trivial. There is not a general pattern to follow when designing an ENM experiment: each species can be specific in terms of habitat and presence/absence information. In [8] the authors report several applications and eventually indicate possible directions for producing robust models. Such directions involve (*i*) improvement of methods for modeling presence-only data, (*ii*) accounting for biotic interactions and (*iii*) assessing model uncertainty. Similar advices come from the BAM diagram for Biotic Interactions described in [17]. Other issues involve the choice of the kind of modeling technique to apply: a model could try to explicitly catch the behavior of a species and its physiological limits and tolerances [4] (*mechanistic* approaches). Otherwise it could automatically extract the correlations between the environmental features vectors and the species presence (*correlative* approaches) [16]. In our methodology a scientist is suggested to use a particular algorithm for simulating the opinion of another biologist on the reviewed map. This algorithm uses a partial mechanistic approach that involves expert knowledge about the species. On the other side we use a correlative technique for generating maps according to different inputs sets.

2.1 ENM Algorithms: Aquamaps and Feed Forward Neural Networks

Several tools allow scientists to produce maps by applying Niche Modeling algorithms [19,5]. In [16] a big collection of techniques is presented along with the kind of scenarios these should be applied to. Artificial Neural Networks (ANNs), in particular, have demonstrated to gain high performances when absence and presence points are available for the species to model. ANNs implement a correlative approach as they try to automatically simulate the probability for a species given certain environmental conditions. Also other models, like SVMs, have gained very good results in ENM [7], but we chose to rely on the advices in [16] and to use ANNs in our method at first stage.

In [18] the authors report a comparison among the most used correlative approaches applied to some marine species of commercial and scientific interest. Among these, the Aquamaps algorithm [13,12] is a presence-only species model, that allows the incorporation of expert knowledge about the species habitat. The Aquamaps distributions are generated using information about species preferences on environmental properties like depth, salinity, temperature, primary production, distance from land and sea ice concentration. Maps are produced at 0.5 degrees resolution. The expert knowledge is used in modeling the habitat parameters and the species preferences. The environmental features values are manually edited before applying the model, then a trapezoidal function is traced for each of them. This function represents the species 'preferred' values for that parameter and can be automatically produced by processing the values ranges associated to the presence points. In particular, the trapezoid is traced on 4 values called *minimum*, *preferred minimum*, *preferred maximum* and *maximum*. These values are calculated, for each parameter, by a rule-based procedure [12] using percentiles of the values observed at the presence points. In some cases the trapezoid can be manually traced by a biologist. The probabilities are produced by multiplying the values of the functions at a certain 0.5 degrees cell in the oceans. Aquamaps presents mechanistic assumptions combined with automatic estimation of parameters values. After the model projection, a scientist can review a map by manually changing the trapezoidal curves or by modifying the values in the produced distribution table. Aquamaps is a reference algorithm for marine species distribution modeling, as it gains high performances if compared to other purely automatic procedures [6]. For such reason we used it for simulating the point of view of another biologist in our method.

2.2 Features Analysis: PCA and HRS

Features analysis is crucial in ENM. A preliminary processing of the features vectors constituting the training set could highlight useless features or could evaluate the potential robustness of the models to produce. One of the most used techniques is the Principal Component Analysis (PCA) [10] a mathematical procedure that aims to reduce the number of dimensions of the features space. PCA uses an orthogonal transformation in the features space for producing independent variables called principal components. This transformation can

be useful for investigating the correlations among the environmental features used in ENM. Adding more dependent variables, in fact, usually does not result in better models. PCA is not specific to biological applications, but other topic-oriented transformations can rely on it. The Habitat Representativeness Score (HRS) [14] is an algorithm based on PCA that applies to marine species environmental features. It measures how much representative sampled habitats are for a certain area of study. HRS has been used for assessing the minimum number of surveys on a study area that are needed to cover a good heterogeneity of species habitat variables. HRS can be applied to two datasets of environmental features, one representing a sampled area and the other a geographical region of interest. A score is produced for each feature, ranging from 0 to 2, with 2 representing completely non-overlapping distributions of values. The lower the HRS the more similar data obtained from a survey to the study area. In this paper we show how HRS can be useful for investigating the robustness of trained models. We applied HRS for assessing how much the features associated to species occurrence points represent a projection area.

3 Methodology

The proposed methodology aims to assist biologists in the evaluation of their manually created maps. The basic assumption is that biologists use their expert knowledge about the species spatial distribution and environmental preferences. This knowledge could be general or specific about some place in which the species lives. In the case of local knowledge it could be that the training set of a model contains indications about a disjoint location. In this case, the biologist and the automatic system start from different assumptions, referring to completely dissimilar environments. This can be generalized by considering the expert knowledge and the automatic system's training set as two sets of multi-dimensional points, where each dimension refers to a separate environmental feature. This leads to one of the following: (*i*) the biologist's knowledge includes the training set, (*ii*) the biologist's knowledge and the training set are disjoint, (*iii*) the biologist's knowledge is completely included in the training set.

The first part of our methodology aims to suggest to biologists in which of these cases the reviewed map could fall. For such aim, they could follow these 6 steps:

1. produce an Aquamaps distribution for the species;
2. review the distribution and produce a *reviewed map*;
3. variate the training set of a Feed Forward Neural Network in order to simulate a spot knowledge or a wide knowledge about the species. Take the best performing network topology in each case;
4. perform a numeric comparison between the reviewed map and the Neural Network maps;
5. use a statistical quality analysis for evaluating similarities among the maps;
6. compare the maps with the Aquamaps distribution as this was the opinion of another biologist.

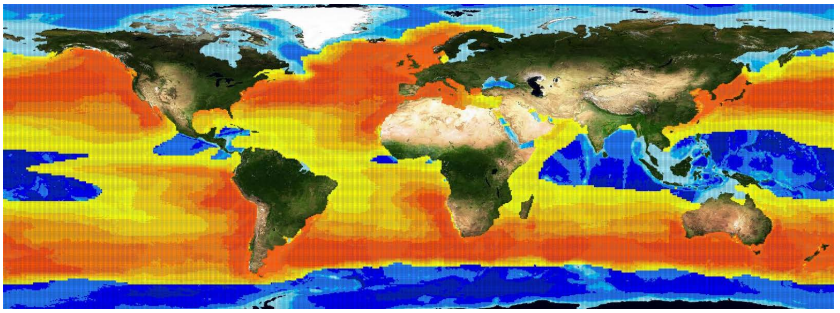


Fig. 1. Manually reviewed Aquamaps distribution for the Basking Shark taken from the AquaMaps website [2]

The second part of the proposed methodology then assesses the reliability of the automatic maps. In order to produce a robust model, the training set has to cover the projection area in terms of the variations in the environmental features values. For example if a Neural Network was trained on a partial set of knowledge (e.g. some points in the Mediterranean sea) but must be projected on a totally different environment (e.g. the Arctic Ocean), then its performances are likely to be unreliable. If a biologist finds the reviewed map to be similar to a possibly unreliable map then he/she might consider to revise it. The final steps in our method can be resumed as follows:

1. apply the HRS to the training sets used for the Neural Networks respect to the projection area;
2. evaluate, numerically, which is the most representative training set;
3. consider the reliability of the reviewed map by looking at the nearest correlative map, the representativeness of its training set and the simulated opinion of another expert.

In the next subsection we show a use case in which the described methodology has been applied to a manually created map for the Basking Shark.

3.1 Use Case

We performed an experiment on the Basking Shark (*Cetorhinus maximus*) marine species starting from the manually created map reported in figure 1. The AquaMaps website [2] provides (*i*) 449 occurrence points, (*ii*) a manually reviewed map and (*iii*) a map generated by the Aquamaps Suitable algorithm [12]. We used the Aquamaps Suitable distribution as a reference to simulate the opinion of another biologist. Figure 2 depicts the presence data distribution (*Presences* set), while figure 3 depicts the Aquamaps Suitable distribution. The maps illustrate that the Aquamaps and the reviewed map are very different.

Following the approach described in Section 3, we trained a Feed Forward Neural Network with the same environmental information used by the Aquamaps

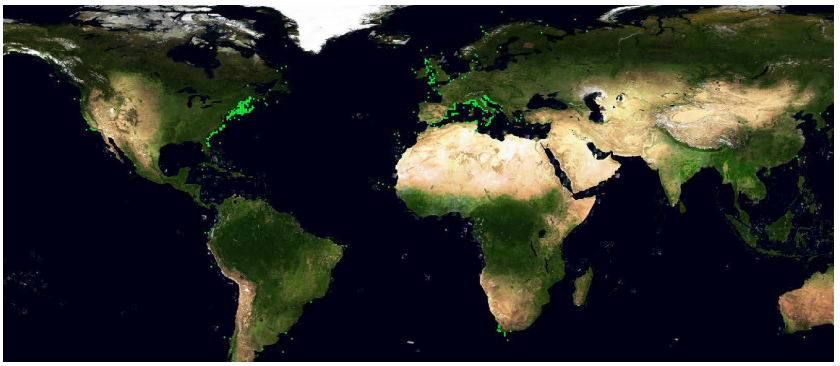


Fig. 2. Presence points for the Basking Shark, provided by Fish-Base [3] and AquaMaps [2]

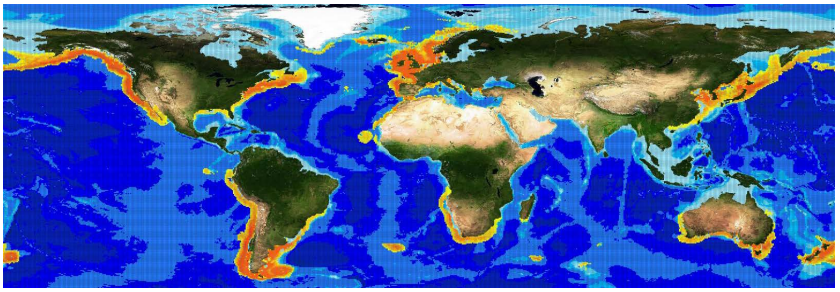


Fig. 3. Aquamaps Suitable range for the Basking Shark

Table 1. Performances of the Neural Networks trained on Dense and Sparse Absences

	Accuracy	Sensitivity	Specificity
NN Dense	64%	96%	33%
NN Sparse	72%	49%	95%

Suitable algorithm. The inputs were vectors of 10 real numbers reporting (for 0.5 degrees oceans cells): the minimum, maximum and mean depth, the mean annual water values for salinity, bottom salinity, surface temperature, bottom temperature, primary production, distance from land and sea ice concentration. Thus, the network had 10 input neurons and 1 output neuron, returning real numbers ranging from 0 to 1. Hidden layers were necessary because the function to be simulated by the network was not linear.

We decided to simulate two variations in the knowledge about the species, by selecting two sets of 449 absence points: (*i*) a *Dense Absences* set, where absence points were close together, (*ii*) a *Sparse Absences* set, where absence points were widely distributed in the oceans. Figure 4 shows the Dense Absences set (left)

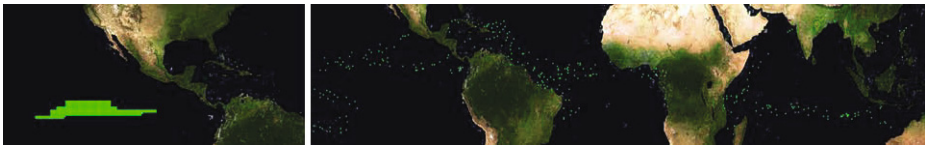


Fig. 4. Dense and Sparse Absences sets for the Basking Shark

and the Sparse Absences set (right). Absence points were necessary because in our case Neural Networks required both positive and negative cases. We chose to variate the absence points sets because presence points are usually reliable information. Reliable absences are much more difficult to find, and it is likely that biologists rely on undocumented knowledge about them. Absence points were not available for the Basking Shark on [2], thus we used the reviewed map for generating them. We took locations with probability lower than 0.1 and higher than 0 to simulate places where the Basking Shark has very low probability of occurrence. We trained a Feed Forward Neural Network on the Presences and Dense Absences sets (*NN-Dense*), and another one on the Presences and Sparse Absences sets (*NN-Sparse*). In the training sessions we changed the number of hidden layers and of neurons in each layer. We adopted a *growing* approach, in which we added neurons and layers as far as the performances on the test set increased. Eventually we took the best performing topologies. We wanted the evaluation to be independent on other maps, therefore we used the 80% of the points for training and 20% for testing. The sets were not overlapping and the best performing model was chosen on the basis of the accuracy on the test set. The best NN-Dense model had 2 hidden layers with 100 neurons in the first layer and 2 neurons in the second, while the best NN-Sparse model had 1 hidden layer with 300 neurons. Table 1 reports the performances of the two models. The accuracy indicates the correct classification rate, the sensitivity the true positives fraction and the specificity the true negatives fraction. The accuracies are comparable, but while the NN-Dense model fits better to true positive classifications, the NN-Sparse prefers true negative classifications.

Figure 5 depicts the NN-Dense model projected on the world oceans, while figure 6 shows the map by the NN-Sparse model. The NN-Dense projection is widespread while the NN-Sparse one seems more similar to the Aquamaps Suitable distribution. We evaluated the similarities among the maps by considering as *true positives* the points in which the reviewed map reported at least a 0.8 probability and as *true negatives* the less than 0.1 probability points. The accuracy of NN-Dense respect to the reviewed map was 96.08% while NN-Sparse gained 66.75%. Taking the Aquamaps Suitable map as reference, NN-Dense gained 82.41% while NN-Sparse gained 93.71%. This evaluation can indicate that the NN-Dense model is more similar to the reviewed map while NN-Sparse is more similar to the Aquamaps Suitable map. For biologists this can be crucial information. The method indicates that the reviewed map is similar to another one generated by an automatic model that uses only limited and local knowledge. A wider knowledge is instead in agreement with an expert system (the

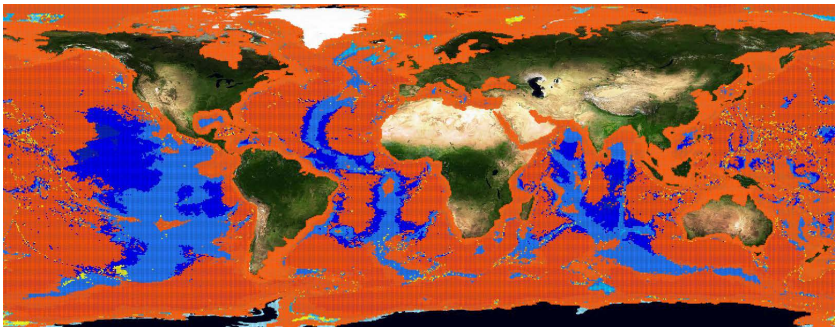


Fig. 5. Neural Network Dense model projected on the world oceans

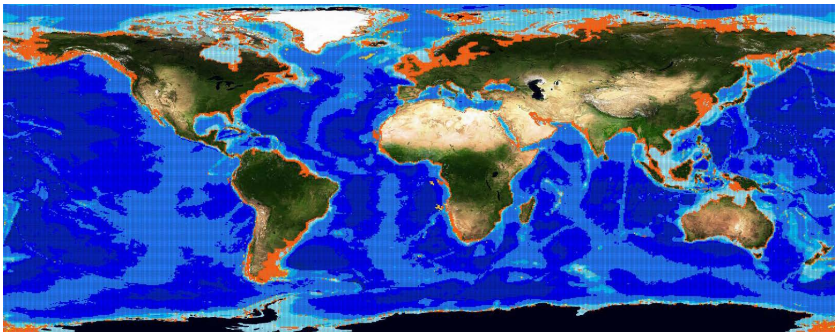


Fig. 6. Neural Network Sparse model projected on the world oceans

Table 2. Overall Habitat Representativeness Score between the projecting area and the training sets

	HRS on world oceans
Presences + Dense Absences	4.49
Presences + Sparse Absences	3.39

Aquamaps Suitable distribution). At this point the biologists choose if the reviewed map has to be revised or if both the expert and the automatic systems are wrong.

As second step, we applied the Habitat Representativeness Score (HRS) technique between the projected area and the presences and absences sets. An overall score was calculated by summing the HRS scores of the involved features. This is different respect to the original procedure. In [14] the author suggests to weigh each score by the inverse of the eigenvalues of the PCA transformation. We avoided this inverse weighting because in our case (*i*) the eigenvalues depended on the ordering of the vectors used for calculating the PCA, (*ii*) we kept all the principal components, because we considered all the features as independent and

equally important, (*iii*) the HRS scores of the parameters were commensurable and an inverse weighting would have given too much importance to less variable dimensions. Table 2 reports the overall HRS of the projection area respect to two combinations of points. As we used 10 environmental features, the maximum (worst) HRS was 20, and the minimum (best) was 0. The combination of Presences and Sparse Absences offer the best performances but according to [14] the score is still high. The maps generated by the Neural Networks might not be robust because the training set does not represent the projection area. On the other side this does not mean they are performing badly, as the scores on the test sets are not low (table 1). At this point the technique offers a more complete scenario about the review: the manual map is similar to a map trained on a local knowledge which is not robust. Furthermore a simulated expert opinion agrees on using a wider knowledge. It is then on the biologists' side the decision to revise or to confirm the reviewed map.

4 Conclusions

We have described a method for using automatic techniques to evaluate a manually reviewed map for a species distribution. The method aims at scientists who want additional tools to revise their maps. We used Feed Forward Neural Networks to simulate different scenarios and calculated the similarity between the processed and the manual maps. By highlighting the similarities a biologist can quickly notice inconsistencies in the manually reviewed map. Furthermore, we used an expert system to simulate the point of view of another scientist that could be taken into account. Finally, by using a PCA based technique we evaluated the reliability of the automatic models by indicating the completeness of the training set. In the Basking Shark experimental case, we demonstrated that the reviewed map was similar to a map trained on a local knowledge and generated by an incomplete training set. Furthermore, by using a more distributed training set with the same dimension, the resulting map was more similar to an expert system's one. This methodology has been adopted in the i-Marine project [1] and is currently used on its stored data sources. Our future activity will concentrate on analyzing the application of our methodology by real users in order to collect more examples and to extend it. Feed Forward Neural Networks may be substituted by better performing models [7] or by *presence points-based* models [18]. Moreover, the current approach can be combined with additional analytical techniques that can reveal further qualitative information about the feature set, and thus improve the reliability of the maps.

Acknowledgments. The reported work has been partially supported by the D4Science-II project (FP7 of the European Commission, INFRA-2008-1.2.2, Contract No. 239019) and by the i-Marine project (FP7 of the European Commission, INFRASTRUCTURES-2011-2, Contract No. 283644).

References

1. The i-Marine European Project (2011), http://www.i-marine.eu
2. Aquamaps Website (2012), http://www.aquamaps.org
3. Fish Base: Searchable global database (2012), http://www.fishbase.org
4. Chuine, I., Beaubien, E.: Phenology is a major determinant of tree species range. Ecology Letters 4(5), 500–510 (2008)
5. Coro, G.: Ecological modelling library for gcube vre. Software (2011), (Sofware) Release 1.0.0 (May 18, 2011)
6. Corsi, F., de Leeuw, J., Skidmore, A.: Modeling species distribution with gis. Research Techniques in Animal Ecology, pp. 389–434. Columbia University Press, New York (2000)
7. Drake, J.M., Randin, C.: Modelling ecological niches with support vector machines. Journal of Applied Ecology 43(3), 424–432 (2006)
8. Elith, J., Leathwick, J.: Species distribution models: ecological explanation and prediction across space and time. Annual Review of Ecology, Evolution, and Systematics 40, 677–697 (2009)
9. Guisan, A., Zimmermann, N.E.: Predictive habitat distribution models in ecology. Ecological Modelling 135(2-3), 147–186 (2000)
10. Jolliffe, I.: Principal component analysis. Wiley Online Library (2005)
11. Kamino, L., Stehmann, J., Amaral, S., De Marco, P., Rangel, T., de Siqueira, M., De Giovanni, R., Hortal, J.: Challenges and perspectives for species distribution modelling in the neotropics. Biology Letters 8(3), 324–326 (2012)
12. Kaschner, K., Ready, J.S., Agbayani, E., Rius, J., Kesner-Reyes, K., Eastwood, P.D., South, A.B., Kullander, S.O., Rees, T., Close, C.H., Watson, R., Pauly, D., Froese, R.: AquaMaps: Predicted range maps for aquatic species (2008), http://www.aquamaps.org/
13. Kaschner, K., Watson, R., Trites, A.W., Pauly, D.: Mapping world-wide distributions of marine mammal species using a relative environmental suitability (RES) model. Marine Ecology Progress Series 316, 285–310 (2006)
14. MacLeod, C.: Habitat representativeness score (hrs): a novel concept for objectively assessing the suitability of survey coverage for modelling the distribution of marine species. Journal of the Marine Biological Association of the United Kingdom 90(07), 1269–1277 (2010)
15. Owens, H.L., Bentley, A.C., Peterson, A.T.: Predicting suitable environments and potential occurrences for coelacanths (latimeria spp.). Biodiversity and Conservation 21, 577–587 (2012)
16. Pearson, R.G.: Species distribution modeling for conservation educators and practitioners (2012), synthesis. American Museum of Natural History, http://ncep.amnh.org
17. Peterson, A., Soberon, J., Pearson, R., Anderson, R., Martinez-Meyer, E., Nakamura, M., Araujo, M.: Ecological Niches and Geographic Distributions (MPB-49), vol. 49. Princeton University Press (2011)
18. Ready, J., Kaschner, K., South, A.B., Eastwood, P.D., Rees, T., Rius, J., Agbayani, E., Kullander, S., Froese, R.: Predicting the distributions of marine organisms at the global scale. Ecological Modelling 221(3), 467–478 (2010), http://www.sciencedirect.com/science/article/pii/S030438000900711X
19. de Souza Muñoz, M.E., De Giovanni, R., de Siqueira, M.F., Sutton, T., Brewer, P., Pereira, R.S., Canhos, D.A.L., Canhos, V.P.: openmodeller: a generic approach to species' potential distribution modelling. GeoInformatica 15(1), 111–135 (2011)

Mining the Viability Profiles of Different Breast Cancer: A Soft Computing Perspective

Antonio Neme[1,2]

[1] Complex Systems Group, Universidad Autónoma de la Ciudad de México
San Lorenzo 290, México, D.F. México
[2] Institute for Molecular Medicine, Finland
neme@nolineal.org.mx

Abstract. Cancer cells present several mutations that allow them to grow faster than normal cells, at the time that enables them to avoid apotosis and other control processes. Cancer cell may be affected by synthetic lethality, which refers to the induction of one or more mutations that affect them, but affect normal cells as little as possible. It is one of the goals of bioinformatics to identify synthetic mutations in order to target specific cancers. If synthetic mutations affect several cancer cells, then it is possible that also some normal cells may be affected. In this contribution, we describe a methodology able to identify a small set of those mutations that affect in a differential way several breast cancer lines. Our methodology is an instance of the feature selection problem and based in genetic algorithms for the exploration of the solution space, but guided by mutual information. Our results show that cancer lines can be profiled with only a small subset of mutations from an original list of hundreds of mutations.

Keywords: cancer profiles, neural networks, genetic algorithms, feature selection.

1 Introduction

Cancer is one of the first causes of death in the developed countries, and it is growing in the developing ones [1]. Among the existing treatments to fight cancer, the directed mutation of cancer cells is one of the most promising ones [2]. Cancer cells present several mutations that do not compromise their viability, and lead them to reproduce at higher speeds than normal cells. Also, some of those mutations eliminate apoptosis so cancer cells are not prone to die, or not to do so in a regular way [2]. The aim of molecular and personalized medicine is to generate other mutations on cancer cells that, together with the already existing mutations in them, affect viability [3]. This is called synthetic lethality. The general idea is that those new mutations that compromise cancer cells at the time that do not affect normal cells, are candidate for drug development.

The general assumption is that a mutation caused by the interference of the messenger RNA by means of small interfering RNA (siRNA) in some kinase affects viability by perturbing a relevant pathway [4]. For the different cancer cells,

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 356–365, 2013.

including breast cancer, several mutations are present and several pathways may had suffered perturbations. By creating new mutations through interference, scientist expect to decrease the viability of cancer cells. Any new mutation can also affect normal cells, which is of course undesirable. Thus, scientist are interested in the minimum set of perturbations (mutations) that may kill cancer cells (see fig. 1-a)).

Those new mutations can be inflected by siRNA, as they affect the expression of some genes by down(up)regulating them. Scientists have explored several of those mutations aiming to attack cancer cells while affecting as little as possible the normal ones. In order to do so, a first option could be to select the siRNA that presents the highest effect in cancer cell viability. For example, for BT20 cancer line, the targeted kinase that impacts the most is PK11. However, that kinase also affects other cancer lines.

If a mutation affects viability of several cancer lines, then the assumption we follow in this project, is that it is affecting different pathways. That affection may also affect pathways relevant for normal cells. We are interested then in finding mutations (kinase targeting siRNAs) that affect differentially all cancer lines. In fig. 2 it is observed that the siRNA targeting kinase PLK1 affects viability in twelve different cancer lines. By affecting that wide range of cancer cells, from our assumption follows that relevant pathways for normal cells may be also affected. That affection may lead to damage on normal cells and thus constitute a constraint for the use of that kinase as a drugable target.

Also, some of the replicates of the same cancer line are mainly affected by different siRNA. This fact contradicts the presumption that the same targeted kinase may be affected in all replicates (instances of the same class).

We studied the data from [5] which consists of 20 breast cancer lines profiled with 719 kinases and kinase-related genes. Each cancer line was profiled by affecting the kinases and its viability was measured with the Z-score. In short, each kinase was targeted using the biological phenomena of RNA interference (RNAi), which happens when a small RNA (mainly small interfering RNA or siRNA) is attached to a messenger RNA (mRNA) and thus the expression of the associated gene is down(up) regulated. This interference can be though of as a perturbation in the natural pathway in the cell. If that perturbation compromises the cell viability, the cell will die or decrease its activity and a way to measure it is ATP consumption. By screening hundreds of siRNA that targets different kinases scientists can identify which siRNAs (and thus, which genes) affect cancer lines and thus, constitute candidate drugs [4]. Fig. 2 shows a general description of the data mined and reported in this contribution.

In this contribution we intend to answer at least the following questions. How does different breast cancer lines are affected by kinases? A subsequent question arise naturally from the previous ones: Expressed in computational terms: What is the minimum set of kinase-targeting siRNA (variables) able to classify the different cancer lines (classes). This is an instance of the feature selection problem. If we want to rephrase that question in order to highlight its biomedical relevance then we can ask: What small set of kinases presents

a differential effect in the viability of cancer lines as to become a signature of cancer lines? To answer these question, we applied a self-organizing maps as visualization method.

The rest of the contribution goes as follows In section 2 we briefly introduce the tools and algorithms applied for seeking an answer to the mentioned questions. In section 3 we present results in which small sets of kinases are enough to discriminate cancer cell lines, and in section 4 we state some conclusions.

2 Methodology and Tools

Feature selection (FS) is a field in machine learning and data mining which impacts in several fields [6]. Mathematically, it can be stated as follows. Let N be the number of potentially relevant variables or attributes that explain or determine the class c_i to which each analyzed vector x_i belongs to and let V the set of variables or measures take for each vector. It is assumed that the class an object belongs to can be discerned from the attributes that describe that object: $c_i = f(V_i)$ where V_i is the description of the vector i and $V_i = \{v_0, v_1, ..., v_{N-1}\}$ is composed of N measures or variables, and $f(.)$ is a function whose domain is that of the possible classes.

If a subset of the N variables in V, named W $(W \subset V, |V| = N, |W| = n)$ can be identified such that there is a function that correlates the class of an object and its description then those $n < N$ variables are the ones relevant for the classification problem. In mathematical terms, FS tries to identify a set of n variables such that $c_i = g(W_i) = f(V_i)$. Two tasks that are embedded have to be solved: the identification of W and the identification of the function g that classifies vectors from W.

Lets call W the space generated by K attributes from space V. The number of possible spaces W is the number of permutations of K positions available to D different attributes $C(D; K)$. The exhaustive search for the case here presented is prohibitively time consuming for $K > 3$. Thus, a search scheme is needed. We applied an heuristic search method, the genetic algorithm, in order to find at most K attributes from V that generate a space such that the statistical correlation between K and the class is maximum. Here, we will measure that statistical correlation in terms of mutual information. Thus, we aim to find a minimum number of variables such that $MI(K; Class)$ is maximum.

There are three schemes to face the FS task: Filter, Wrapper, and embedded methods [7]. We propose here the use of a wrapper method, and, as described below, the evaluation of the selected features is done by means of a non-linear correlation measure. We propose a genetic algorithm able to identify n variables . As an approximation of the classification function g we propose the use of mutual information (MI). Fig. 1 shows te general algorithm for feature selection. As a variant of the existing FS algorithms, our proposal is based in MI, and the inclusion of variables is probabilistic: the more the MI between a variable and the class, the more probable the variable will be selected to become part of an individual.

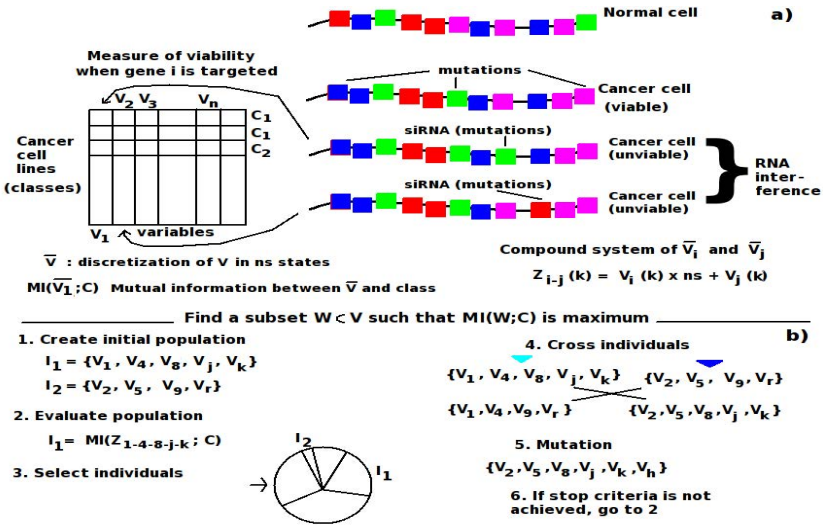


Fig. 1. Feature selection scheme implemented by a genetic algorithm. Each individual represents in its chromosomes the siRNA considered for the complete identification of the cancer class.

For visualizing purposes we applied self-organizing maps (SOM). SOM are able to generate non-linear projections of high-dimensional data into low-dimensional spaces. Those projections are in general good approximations of the data distribution in the high-dimensional space [11].

Fig. 1-b shows the general workflow we followed to identify those features (kinases and kinase-related genes) that differentially affect the cancer lines. Once again, one of our goals is to find a subset of the 719 kinases that differentially affects the 20 different cancer lines. By doing so, our workflow is in a position to suggest to biomedical experts a subset of the studied kinases that may affect specific breast cancer cells.

3 Results

We decided to apply mutual information instead of the more common correlation measures (Spearman, Pearson) as it is able to detect non-linear and subtle correlations [8]. Mutual information between two random variables X and C is expressed as $MI(X;C)$, where X is in this work refers to one of the siRNA measures of viability, and C is the class or cancer cell line MI is defined as:

$$MI(X;Z) = \sum_{i}^{\#statesinX} \sum_{j}^{\#statesinC} P(i,j) log \frac{P(i,j)}{P(i)P(j)} \quad (1)$$

For the case of classification, the number of states in C is just the number of classes, and ns is the number of states in X. If X is a continuous variable, then it can be discretized into ns different states. In information-theoretic algorithms the guiding quantity to be minimized is the Renyi Quadratic Entropy of the error between the output of the system and the actual label [13].

In general, for artificial datasets with no noise, all entropy in the label can be removed from the list of attributes $\bar{X}$ that define the high-dimensional feature space. That is, $MI(\bar{X}; C) = H$. Mutual information between the compound system of all attributes or variables and C ($MI(\bar{X}; C)$) tends to disipate all entropy in the label. That is, when $ns \to \infty$, $MI(\bar{X}; Z) \to H$. When the guiding quantity of an adaptive system F (such MLP) is the mutual information, we are considering high-order momentum able to capture non-linear correlations in data [9,8].

In the analyzed dataset, there are 59 vectors belonging to 20 different classes. The entropy of the dataset is $H = 4.14$ bits. In fig. 2 it is shown the probability distribution of the number of siRNA and the MI for the class (cancer cell line). MI was obtained in this case with a discretization of viability into 20 states. The maximum MI is 2.7023 obtained between siRNA targeting kinase CHKB. Note that this kinase is not listed as one of the kinases presenting the highest effect for any of the cancer lines (see fig. 2).

Our dataset consists of a high-throughput screening (HTS) of 20 breast cancer cell lines (20 classes). On each screening, 719 kinases were targeted. In a typical HTS, plates are filled with such that on each well of the plate a certain cancer line is present and also one of the 719 siRNA that targets the associated kinase. That is, on each well a siRNA will mutate (affect) the cell, and the impact of that interaction will be measured. If the viability of the cancer cell is compromised, a negative measure is obtained. A measure of -2.0 or less indicates that the cancer cell was not able to survive, and thus, the targeted kinase may be relevant for the cancer cell. Each well on the plate is considered an independent experiment, and hundreds or thousands of simultaneous experiments can be performed. Each experiment was performed two or three times. Thus, for each cancer cell the dataset contains one or two technical replicates. In total, there are 59 input vectors, that correspond to 19 cancer lines with two replicates (three experiments) and one cancer line with only one replicate(two experiments).

In the first set of analysis, the 59 cancer lines were the input vectors, and the measure of viability for each of the 719 siRNA targeting the kinases were considered as the variables. That is, a given cancer line is described as a vector of 719 real values, in which each one of those values represent the impact that the associated siRNA presented on the cell viability.

Fig. 3 shows the map obtained by SOM on a lattice of size 12×12 units for of the 59 cancer cells. Those 59 input vectors are described by 719 measures, which define a 719-dimensional space. It is observed that replicates from the same cancer line tend to be located in the same area, but there are some exceptions. One interpretation of this map is that the 719 siRNA affect in different ways the 59 cancer lines. However, as stated in the introduction, by targeting different kinases, the viability of normal cells may be affected. Thus, we are interested in

Cancer line	Class	Kinase with highest effect on cancer line	Other cancers with same kinase as the most influential
BT20.1	0	AURKA	1
BT20.2	0	PLK1	12
BT20.3	0	PLK1	12
BT474.1	1	COASY	0
BT474.2	1	PLK1	12
CAL120.1	2	MAP4K4	0
CAL120.2	2	PRKAB1	0
CAL120.3	2	PRPS1L1	0
CAL51.1	3	PLK1	12
CAL51.2	3	PLK1	12
CAL51.3	3	PLK1	12
CAMA.1	4	PLK1	12
CAMA.2	4	PLK1	12
CAMA.3	4	PLK1	12
HCC1143.1	5	CHEK1	0
HCC1143.2	5	GUCY2D	1
HCC1143.3	5	CHEK1	0
HCC202.1	6	PLK1	12
HCC202.2	6	PLK1	12
HCC202.3	6	PLK1	12
HS578T.1	7	PIK3C2A	0
HS578T.2	7	GUCY2D	1
HS578T.3	7	GUCY2D	1
JIMT.1	8	PLK1	12
JIMT.2	8	PLK1	12
JIMT.3	8	PLK1	12
MCF7.1	9	FYN	0
MCF7.2	9	FYN	0
MCF7.3	9	FYN	0

Cancer line	Class	Kinase with highest effect on cancer line	Other cancers with same kinase as the most influential
MDAMB134.1	10	AURKA	1
MDAMB134.2	10	AURKA	1
MDAMB134.3	10	AURKA	1
MDAMB157.1	11	PLK1	12
MDAMB157.2	11	PLK1	12
MDAMB157.3	11	PLK1	12
MDAMB231.1	12	MPP3	0
MDAMB231.2	12	PKLR	0
MDAMB231.3	12	FN3KRP	0
MDAMB453.1	13	PLK1	12
MDAMB453.2	13	PLK1	12
MDAMB453.3	13	PLK1	12
MDAMB468.1	14	PRKCG	0
MDAMB468.2	14	PLK1	12
MDAMB468.3	14	PLK1	12
SKBR3.1	15	PLK1	12
SKBR3.2	15	PLK1	12
SKBR3.3	15	PLK1	12
SUM149.1	16	JAK2	0
SUM149.2	16	JAK2	0
SUM149.3	16	JAK2	0
SUM44.1	17	STK11	0
SUM44.2	17	STK11	0
SUM44.3	17	PLK1	12
T47D.1	18	EPHA6	0
T47D.2	18	EPHA6	0
T47D.3	18	EPHA6	0
VP229.1	19	DCK	0
VP229.2	19	PLK1	12
VP229.3	19	CDK4	0

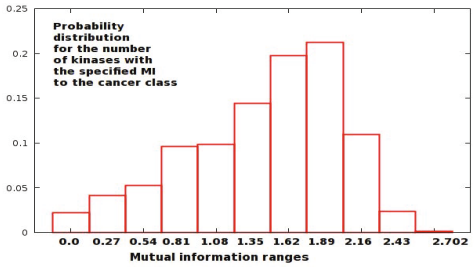


Fig. 2. Kinases with the highest effect on cancer lines. For each of the replicates it is shown the most influential targeted kinase. It is observed that replicates of the same cancer line may present different most influential kinases. Also, one kinase is the ost influential in 12 different cancer lines. b) Probability distribution of mutual information between kinases and class. One kinase presents MI in the range 2.43 – 2.702, whereas four kinases present a MI below 0.27.

selecting some of those 719 siRNA that are still able to target in a differential fashion the 20 cancer cells.

The data reflecting the siRNA effect can be filtered more, as values above -1 mean that there was not effect on the cell viability. From the original dataset, we constructed a second one in which siRNAs that do not present a value of -1 or less for at least 15 of the 59 cancer lines are filtered out. This reduced dataset contains 25 siRNA and thus, cancer profiles are located now in a 25-dimensional space. Fig. 4 shows the SOM formed over a 12x12 lattice.

Again, it is observed in fig. 4 that the selected 25 siRNA have a more or less differential influence in the cancer lines. It is observed that some cell lines are very similar at the time that replicates of the same cancer are not as similar as expected. We are interested in finding a subset of those 25 variables that can determine with less error the cancer cell. That is, we want to select features from the list of 25 selected siRNA that have effect in the majority of the cancer lines.

A näive method to select $n < N$ siRNA from the list of the filtered $N = 25$ variables would be to select the n with the highest correlation measure. That

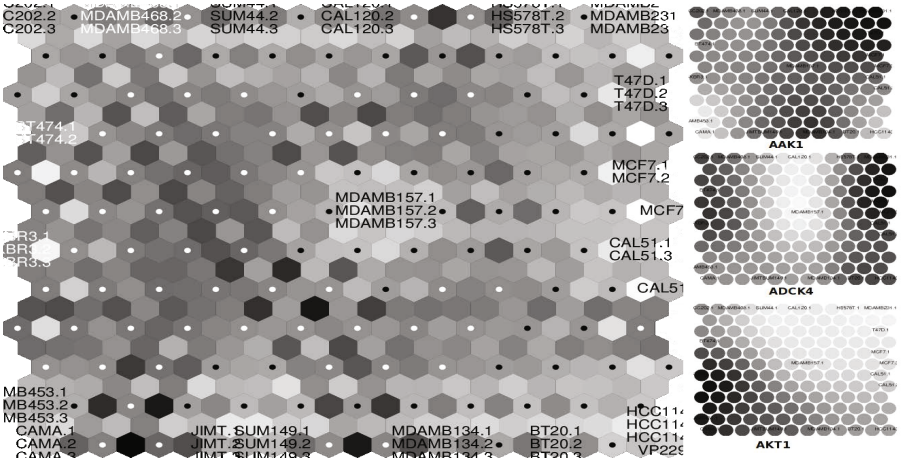


Fig. 3. SOM for the 20 breast cancer lines. There are three instances for each one of the 20 cancer types, except for BT474, which only presents two instances. Each cancer line is described by 719 variables, that represent the viability on the cell when targeting independently 719 kinases or kinase-related genes. The measures (planes) for three siRNA are shown. It is observed that these three siRNA have a differential impact in the different cancer lines.

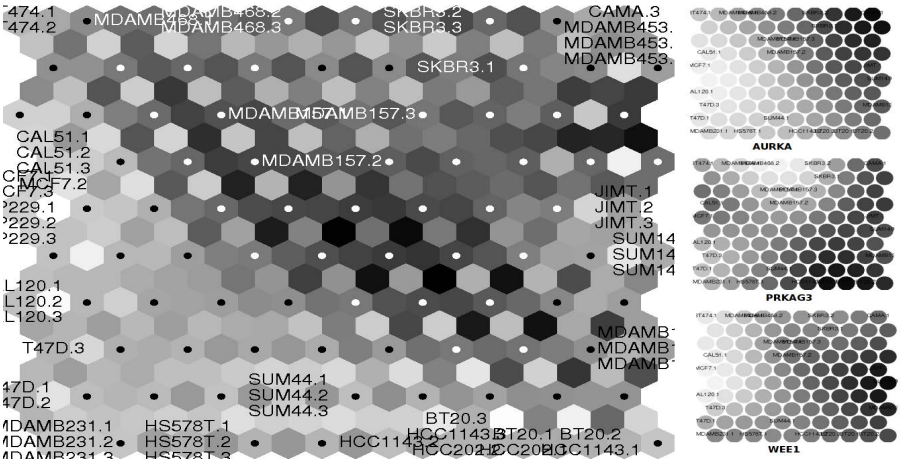


Fig. 4. SOM for the 20 breast cancer lines. There are three instances for each one of the 20 cancer types, except for BT474, which only presents two instances. Each cancer line is described by 25 variables, that represent the viability on the cell when targeting independently 25 kinases or kinase-related genes. The measures (planes) for three siRNA are shown.

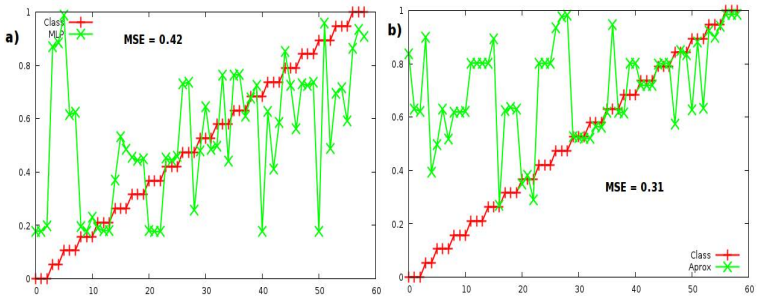


Fig. 5. a) The estimated class from the 7 variables (siRNA) with the highest mutual information. MLP was of 7 inputs, two hidden layers, each one with 10 neurons (that was the combination with best results). MLP was trained with genetic algorithms for 5000 epochs and with a population of 200 individuals.b) The estimated class fro the 7 variables (siRNA) selected by the FS algorithm.

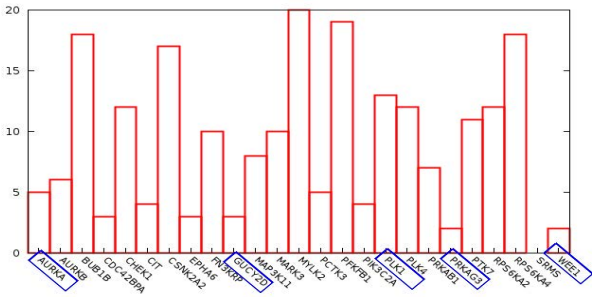


Fig. 6. The number of times each one of the 25 siRNA were selected by the FS algorithm. The five most informative variables, accordingly to equation 1, are marked in blue.

measure can be Spearman, Pearson or MI, but that selection criteria is a poor decision as it does not take into account dependencies between variables. A methodology to establish if a class can be inferred from the existing variables is to use an universal approximation formalism. We applied multilayer perceptrons (MLP) in order to compute the class (cancer line) from the specified siRNAs values. Fig. 5-a shows the predicted class from the $n = 7$ variables with the highest MI to the class (class is normalized). The approximation task was obtained with a MLP with two hidden layers and several number of units on each layer were tested. MLP was trained with a genetic algorithm for 5000 epochs, 200 individuals and a probability of mutation of 0.05. It is presented the best case, and it is observed that the approximation is not a good one (MSE = 0.4).

In contrast with the results achieved when a näive scheme is followed, we present in fig. 5-b the predicted class when $n = 7$ variables were selected following the FS algorithm depicted in the previous section. Again, the MLP was

trained by genetic algorithms for 5000 epochs with a population of size 200 and probability of mutation of 0.05. The error is lower for

We ran 200 times the FS algorithm and we present in fig. 6 the probability for each siRNA of being part of the subspace. It is observed that the most informative variables, shown with a surrounding blue box, are not the most selected features. The fact that variables not as informative as those are selected ost frequently is explained by the fact that those less informative variables, when combined with other variables, may offer more information. That is exactly what the proposed FS algorithm does.

Counting with a small number of variables (< 719) that are enough to classify the cancer cell lines is relevant because those few variables conform a manageable profile. With that profile a detailed biological explanation is more accessible. From this point, the biological relevance of the siRNA selected by the proposed algorithm can be tracked.

4 Conclusions

The identification of biological variables that impact in a given process is a relevant task. Here, we have described a workflow that was able to identify a set of small interfering RNA affecting differentially a group of 20 cancer lines. The relevance of identifying that small subset of variables is not only a computational benchmark, but also, it pinpoints specific genes for further study.

We proposed a feature selection algorithm based on evolutionary computation and guided by mutual information. The algorithm was applied with success in real data from the cancer biology field.

Acknowledgments. A. Neme thanks ICyTDF for a postdoctoral fellowship. Also, A. Neme is in the SNI - CONACYT México.

References

1. WHO Disease and injury country estimates. World Health Organization (retrieved October 4, 2009)
2. Hanaha, D., Weinberg, R.: Hallmarks of Cancer: The Next Generation. Cell 144, 646–674 (2011)
3. Ebert, M.S., Sharp, P.A.: Roles for microRNA in conferring robustness to biological processes. Cell 149 (2012)
4. Volinia, et al.: Reprogramming of miRNA networks in cancer and leukemia. Genome Research 20, 589–599 (2010)
5. Brough, R., Frankum, J., Sims, D., et al.: Functional viability profiles of breast cancer. Cancer Discovery 1, 260–273 (2011)
6. Guyon, I., Elisseeff, A.: An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3, 1157–1182 (2003)
7. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics 23(19), 2507–2517 (2007)

8. Cellucci, C.J., Albano, A.M., College, B., Rapp, P.: Statistical Validation of Mutual Information Calculations: Comparison of Alternative Numerical Algorithms. Physical Review E 71(6) (2005), doi:10.1103/PhysRevE.71.066208
9. Silva, L., Marques de Sá, J., Alexandre, L.: Neural Network Classification using Shannon's Entropy. In: Proceedings of the 13th European Symposium on Artificial Neural Networks Bruges, Belgium, April 27-29 (2005)
10. Shannon, C.A.: Mathematical Theory of Communication. Bell System Technical Journal 27, 379–423, 623–656 (1948)
11. Kohonen, T.: Self-organizing maps, 2nd ed. Springer (2000)
12. Yin, H.: The Self-Organizing Maps: Background, Theories, Extensions and Applications. In: Fulcher, J., Jain, L.C. (eds.) Computational Intelligence: A Compendium, vol. 115, pp. 715–762. Springer, Heidelberg (2008)
13. Santos, J., Marques de Sá, J., Alexandre, L., Sereno, F.: Optimization of the error entropy minimization algorithm for neural network classification. ANNIE vol. 14 of Int. Eng. Sys. Through Art. Neural Net, pp. 81–86. ASME Press, USA (2004)
14. Cortes, M.L., Ruiz-Shulcloper, J., Alba-Cabrera, E.: An overview of the evolution of the concept of testor. Pattern Recognition 34, 753–762 (2001)

Image Representation and Processing Using Ternary Quantum Computing

Simona Caraiman and Vasile Manta

Computer Engineering Department, Technical University of Iasi,
D. Mangeron 27, 700050 Iasi, Romania
{sarustei,vmanta}@cs.tuiasi.ro

Abstract. In this paper we address the recently outlined field of Quantum Image Processing and propose a novel model for representing a quantum image. In our approach we use multilevel quantum systems to store and process images because of their advantages in terms of dimension of the available Hilbert space, computational power, physical implementation and security of quantum cryptographic protocols. In particular, we focus on the quantum image representation using qutrits (3-level quantum systems) and discuss possible implementations for basic image processing tasks such as image complement, image binarization and histogram computation.

Keywords: Quantum Information, Image Processing, Multilevel Quantum Logic, Qutrits.

1 Introduction

The quantum technology seems to be one of the most promising technologies for building future computing systems. A direct consequence of the principles of quantum physics is the immense computing power of a quantum machine compared to that of a classical one. This is due to three remarkable quantum resources that have no classical counterparts: quantum parallelism, quantum interference and entanglement of quantum states. The remarkable properties of quantum systems recently led to the emergence of innovative ideas in all major fields of computing, including graphics processing and most of them are based on two major results: Shor's factorization algorithm [1] and Grover's search algorithm [2].

As for Quantum Image Processing, the research in the field has encountered fundamental difficulties as it is still in its infancy. The confronted problems refer to the mechanisms of representing and storing an image on a quantum computer and to the basic preparation and processing operations. Concepts of quantum image representation have been proposed: Qubit Latice [3, 4], Real Ket [5], FRQI [6] and also a method for storing and representing binary geometrical forms [7].

It has also been proven that there are quantum processing transformations more efficient than their classical versions: quantum Fourier transform, quantum wavelet transform [8] and the quantum cosine transform [9]. Using these efficient

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 366–375, 2013.

operations in image processing could prove the potential of the quantum image processing field. Still, there are fundamental differences between classical and quantum operations, the latter being necessarily invertible due to the reversible nature of quantum computation. Therefore there are classical image processing operations such as convolution and correlation that cannot be applied to quantum images [10].

All these ideas are formulated using the binary quantum logic which is consistent with most approaches to quantum computing. Nevertheless, recent studies have indicated several advantages in expanding the quantum computer from qubits (the binary unit of information in quantum logic) to multilevel systems, named qudits. It seems that currently, the major obstacle in quantum computing is posed by the small limit of coupled quantum bits realizable in a practical physical system. The use of higher-dimensional quantum states enables an exponential increase of the available Hilbert space with the same amount of physical resources. Moreover, some of the quantum computational technologies proposed so far, such as ion traps or quantum dots, are not intrinsically binary. Thus, in practice, qubits are often realized in higher dimensional physical systems by truncating excess levels. In particular, ternary quantum systems offer several benefits over binary and other multilevel quantum systems, such as enhanced security in some quantum information processing protocols [11–13], optimization of Hilbert space dimensionality of an entangled quantum system [14] and more efficient logical gates implementation [15, 16]. Also Bell inequalities were developed for arbitrarily high-dimensional systems [17] and it seems that the systems of entangled qutrits are expected to produce larger violations of the nonlocality principle [18]. Moreover, the experimental realizations of multilevel quantum systems that have been recently reported prove that the physical implementation of high-dimensional quantum logic is feasible[16, 19–22].

In this paper we investigate the use of ternary quantum logic for representing a quantum image and provide ternary quantum circuits for basic image processing tasks such as image complement and binarization. We also describe a procedure for computing the histogram of a quantum image and discuss the advantages of using ternary instead of binary quantum computing systems for such a task. We show that these advantages mainly arise from the increase in state space and in precision of ternary variants for the quantum Fourier transform, Grover's operator and phase estimation.

2 Preliminaries - Quantum Information and Quantum Computation

2.1 Binary Quantum Logic

The unit of information in binary quantum logic is the *qubit*, the quantum analogous of the classical bit. In Dirac notation, the state of a qubit can be completely described by the superposition of two orthonormal basis states, labeled $|0\rangle$ and $|1\rangle$ (in a Hilbert space $\mathcal{H} = C^2$, $|0\rangle = (1 \quad 0)^T$, $|1\rangle = (0 \quad 1)^T$). Any state $|\Psi\rangle$ can be described as a linear combination of the basis states:

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad |\alpha|^2 + |\beta|^2 = 1, \tag{1}$$

where α and β are complex numbers and represent probability amplitudes with respect to the $\{|0\rangle, |1\rangle\}$ computational basis. When measuring a qubit either the result 0 is obtained, with probability $|\alpha|^2$, or 1 with probability $|\beta|^2$.

A collection of n qubits is called a quantum register with dimension n. The general state of a n-qubit register is

$$|\Psi\rangle = \sum_{i=0}^{2^n-1} a_i|i\rangle, \tag{2}$$

where $a_i \in \mathcal{C}$, $\sum_{i=0}^{2^n-1} |a_i|^2 = 1$. This means that the state of a n-qubit register is represented by a complex unit vector in Hilbert space $\mathcal{H}_{2^n}$.

To achieve a coherent computation, the quantum registers need to be isolated such that there is no interference with the environment. Changes in the state of the system should be produced adiabatically which implies that all computation processes have to be reversible. Any reversible operation can be described by a unitary operator, U, with $U^{-1} = U^*$. The composition of unitary operators is also unitary because $(UV)^{-1} = V^*U^*$. A general unitary transformation, U, in the bi-dimensional Hilbert space C^2 can be represented by gates like in classical circuits. Elementary quantum gates include single qubit gates (Not, Hadamard, phase-shift, rotations), controlled gates (C-Not, C-Phase) and the Toffoli gate. For a quantum register with several qubits, the evolution of the system can be described using the tensor product of single qubit unitary operators.

The only irreversible operation allowed in a quantum computational process is the measurement operation. Measuring the state of n qubits reduces the dimensionality of the $\mathcal{H}$ space with a factor of 2^n. In other words, after measurement the system collapses to one of the basis states and the values of the probability amplitudes cannot be recovered.

2.2 Ternary Quantum Logic

In multi-valued quantum logic, the unit of information is the *qudit*. A 3-level qudit, called *qutrit*, is implemented by a quantum system having three mutually orthogonal states $|0\rangle$, $|1\rangle$ and $|2\rangle$ that form a basis in the Hilbert space C^3. The superposition state of a qutrit can be described by

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle + \gamma|2\rangle, \quad |\alpha|^2 + |\beta|^2 + |\gamma|^2 = 1, \tag{3}$$

where α, β and γ are the complex probability amplitudes of the basis states and $|0\rangle = (1 \quad 0 \quad 0)^T$, $|1\rangle = (0 \quad 1 \quad 0)^T$, $|2\rangle = (0 \quad 0 \quad 1)^T$. Analogous to the binary quantum system, a measurement of the qutrit yields $|0\rangle$ with probability $|\alpha|^2$, $|1\rangle$ with probability $|\beta|^2$ and $|2\rangle$ with probability $|\gamma|^2$.

An n-qutrit quantum system can be represented by a superpositon of 3^n basis states, thus a quantum register of size n can hold 3^n values simultaneously, whereas an n-qubit register can only hold 2^n values.

In order to build quantum circuits for implementing ternary operations, a set of universal gates is required. Even though multi-level quantum computing is a new field, there are several works done on developing multi-level and specifically ternary quantum logic synthesis methods. There are several proposals for universal multi-level quantum gates: two qudit Muthukrishnan-Stroud (M-S) gates together with arbitrary one-qudit gates [23], control gates [24–27], Ternary Swap, the Ternary NOT and the Ternary Toffoli [28], generalized ternary gates and projection operations [29]. Minimization of gate cost was shown to be achievable using quantum multiplexers and the method of iterative deepening depth first search [30] .

3 Representation of Quantum Images

In order to represent the quantum image we will use a quantum register prepared in the state

$$|I\rangle = |C\rangle \otimes |\psi\rangle. \tag{4}$$

This quantum state integrates both color and position information. Pixel positions are coded in $|\psi\rangle$ using $2n$ qutrits if we consider, without loss of generality, that there are $N = 3^n \times 3^n$ pixels in the image.

In previous work regarding quantum image representation the color information is encoded using a single qubit [4], [6]:

$$|\phi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\gamma}\sin\frac{\theta}{2}|1\rangle. \tag{5}$$

The real parameter θ encodes the frequency of the electromagnetic wave while γ is left uninitialized. We could employ a similar representation using qutrits, but the problem with this approach is that such a protocol produces general quantum states. Even though the color of a pixel is not necessarily a basis state, the visual information can be accurately retrieved using a statistical procedure involving multiple measurements of identically prepared states. Nevertheless, as discussed in the final section, we envision the application of quantum amplitude amplification in order to achieve image processing tasks such as histogram computation. In order to apply such a mechanism, the marked states need to be computational basis states, so we find such a representation of the quantum image unsuitable for this purpose.

In order to overcome this problem we propose that $m = \log_3 L$ qutrits are used to encode the L gray levels present in the image, meaning that C is an m-qutrit register. In comparison to using qubits for implementing the proposed representation scheme the state space is increased by a factor of $(\frac{3}{2})^m$ since the Hilbert space of m qutrits has the same dimensionality as $m \log_2 3$ qubits. Even though the necessary state space is larger than in the representation suggested in [4], the possibilities for processing operations are much more valuable. Moreover, a partial state space recoup is achieved by qutrit codification of pixel positions.

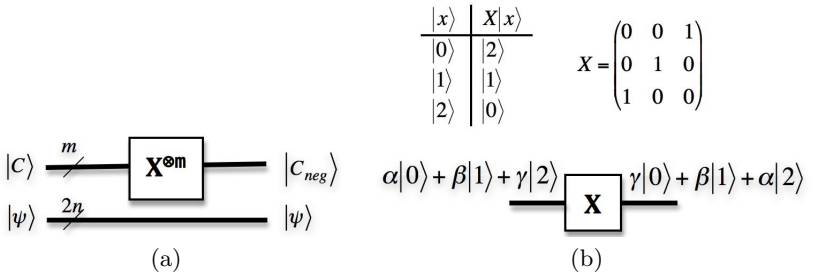


Fig. 1. (a) Quantum circuit implementing the image negative. (b) Truth table, matrix and graphical representation for the action of the ternary NOT gate.

4 Quantum Image Processing

4.1 Complement of a Quantum Image

The complement, or the negative, is a basic image processing operation that comes handy when trying to enhance white or gray detail embedded in dark regions of an image, especially in the case of images with dominant black areas. It can be achieved with the following transform:

$$s = T(r) = L - 1 - r, \tag{6}$$

where r and s are the pixel values before and after processing, respectively.

A quantum circuit that achieves this operation can be built using a ternary NOT gate as described in Fig. 1(a). The truth table and graphical representation of the ternary NOT gate are presented in Fig. 1(b). This gate is equivalent to the $Z_3(02)$ one-qutrit permutative gate introduced in [26].

4.2 Image Binarization

Even though binarization is a basic image processing task, it is of great importance due to its extensive use in the segmentation process. It has the purpose of transforming a gray level image in a binary one which has only two possible values for each pixel, typically black and white. This transformation uses a threshold value and classifies all pixels with values above the threshold as white and all the other pixels as black:

$$s = T(r) = \begin{cases} 0, & 0 \le r \le thres \\ L-1, & thres < r \le L-1 \end{cases}. \tag{7}$$

In order to implement the binarization operation a quantum gate formalism needs to be described for the $if - then - else$ conditional structures. In binary quantum logic such structures can be built out of NOT and controlled-U gates. Quantum controlled gates use a control input to determine whether a specific unitary action is applied to a target input.

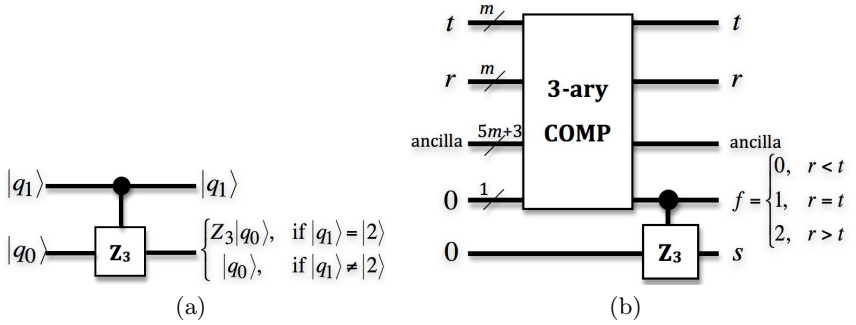


Fig. 2. (a) Graphical representation for the action of the M-S gate. (b) Ternary quantum circuit for implementing the image binarization operation. The ternary quantum comparator has a quantum cost of $56n + 6$ [31]. The output f of the comparator is used as a control qutrit in the M-S gate that complements the output pixel value, i.e. here the Z_3 gate is the X gate presented in Fig. 1(b).

A basic version of the controlled-U operation is given by the M-S gate [23], which is the generalized ternary correspondent of the binary CNOT gate. Its quantum cost is assumed to be 1 and its action is described in Fig. 2(a). The control input, $|q_1\rangle$, passes unchanged while the target qutrit, $|q_0\rangle$, is transformed with a one-qutrit permutation gate, Z_3, if $|q_1\rangle = |2\rangle$. Otherwise it also passes unchanged.

In the case of binarization the conditional operation is controlled by an inequality. A quantum circuit implementing a conditional operation controlled by a 'greater-than' expression can be built using a quantum comparator and a M-S gate.

The design of ternary quantum comparator circuits has been addressed in [32] and [31]. Individual quantum circuits for 'less-than', 'equal' and 'greater-than' were proposed in [32], based on 1-qutrit, 2-qutrit and M-S gates. On the other hand, a single circuit that outputs all the three comparison results at once was described in [31]. It has a lower quantum cost and uses fewer ancilla qutrits by exploiting a ternary quantum full adder, 1-qutrit permutation gates, M-S and 3-qutrit Toffoli gates.

The resulting image will be stored in an additional output register, initially prepared in state $|0\rangle$. Because the output image is binary, a single qutrit per pixel is enough to hold the color information.

Using the quantum comparator described in [31] the quantum binarization circuit is presented in Fig. 2(b) and has the following behavior:

$$U_{bin}|t\rangle|r\rangle|0^{\otimes 5m+3}\rangle|0\rangle|0\rangle = |t\rangle|r\rangle|\psi\rangle|f\rangle|s\rangle, \tag{8}$$

where r is the input gray level, s is the output pixel value, t is the gray level used as threshold, $|f\rangle$ is the comparator output and $|\psi\rangle$ is a $5m+3$ qutrit ancilla register required by the comparator. The quantum cost of the binarization circuit is given by the comparator cost plus the final M-S gate, i.e. a total of $56m + 7$.

4.3 Histogram Computation

The histogram of an image represents the relative frequency of occurrence of the various colors (gray levels) in the image. Mathematically speaking, for a digital image with gray levels in the range $[0, L-1]$, the histogram is a discrete function

$$h(g_k) = \frac{n_k}{N}, k = 0, 1, ..., L-1, \tag{9}$$

that is the number of pixels n_k having greyscale intensity g_k as a fraction of the total number of pixels N.

In the histogram definition given above it is assumed that each possible intensity value of the image corresponds to a single bin of the histogram. For purposes of computing the histogram when there are many possible color values, several values can be grouped into a single bin. A classical procedure for histogram computation involves initializing the bins of the histogram to zero and then computing the values associated to each bin by accumulation.

A quantum variant for the histogram computation can be devised if we reinterpret it in the terms of a search problem with multiple solutions. The value associated with a bin is obtained by searching the image pixels having a certain color and counting the solutions. In this case the quantum counting algorithm can be used for each bin thus providing a quadratic speedup over the classic procedure as the actual retrieval of the solutions to the search problem, i.e. the pixels with a certain color, is not necessary.

The quantum counting algorithm is an inspired combination of quantum search with the quantum Fourier transform that appears in the eigenvalue estimation problem. Specifically, the quantum counting algorithm exploits the fact that the eigenvalues of Grover's amplitude amplification operator are related to the number of solutions, t, to the search problem [33]. This allows the formulation of an algorithm that approximately counts the solutions of the search problem in $O(\sqrt{N})$ time, with a success probability of at least $8/\pi^2$, where N represents the dimension of the search space. In the quantum counting algorithm we must pick the number of bits of precision to which we wish to estimate the eigenvalues (and hence the precision with which we can estimate the number of solutions). Various relations that bound the accuracy with which the number of solutions can be estimated are summarized in [34].

Ternary variants for the quantum Fourier transform are proposed in [23], [35], [36]. Besides the state space increase, compared to traditional binary QFT, the ternary transform also improves approximation properties by a factor of $(\frac{3}{2})^n$ [36]. Generalizations of Grover's search algorithm to multi-value logic were proposed in [37] and [38]. Both approaches allow for a reduction in memory register size by using fewer qudits to represent more information states. Moreover, the method described in [38] needs fewer iteration steps to find the marked items compared to the qubits encoding procedure, thus leading to less dissipations and errors in the physical systems. Also, the authors describe how such a scheme could be achieved efficiently within current state-of-the-art technology.

The strategy for eigenvalue estimation that appears in the quantum counting algorithm is to create a quantum state, called the 'phase state', in which the

desired eigenvalue appears as a phase factor and to extract this phase factor with the help of a technique called quantum phase estimation. Thus, estimating the eigenvalues of a quantum operator reduces to a two-step process of creating the special phase state and then performing phase estimation on it [34]. A multivalued quantum logic version for the phase estimation problem was developed in [39]. When compared to the binary quantum logic version, the multivalued algorithm turns out to be more robust, with a significant decrease in the number of required qudits and with a drastic improvement in the precision and success probability.

5 Conclusions and Future Work

In this work we have suggested a new model for representing quantum images and provided an insight on how to realize image processing tasks. Our method exploits the advantages of ternary quantum systems and allows for a significant decrease of the state space needed to represent quantum images while providing the means for faster and more efficient operations. We have shown that implementing the image complement and image binarization operations can be achieved using elementary ternary quantum gates such as 1-qutrit permutation gates, Muthukrishnan-Stroud and Toffoli gates.

Extending the results obtained in this paper to other image processing tasks is desirable and would contribute further to investigating the feasibility of implementing the quantum computing paradigm using higher dimensional quantum systems. One example would be to explore a similar approach for expressing other histogram processing operations such as histogram equalization. Threshold-based segmentation would then be one step closer to being formulated using the ternary quantum logic.

As pointed out in the first section of the paper, systems of qutrits have several advantages over higher dimensional ones. One such advantage relies in the high degree of entanglement that can be produced between these particular systems. A natural development would be to extend the results obtained in [3] by investigating how qutrit entanglement could be exploited in order to enhance quantum image processing tasks.

Even though both Quantum Image Processing and Multilevel Quantum Logic are still in an early stage of development, blending these two fields could result in interesting ideas for the advent of the main framework of Quantum Information Processing. This would have a significant impact on the exploration of theoretical concepts of quantum information, the design of new quantum algorithms and the development of novel applications for quantum information processing. It could also represent an overall justification for the immense efforts needed for building working quantum computers.

Acknowledgements. This paper was supported by the project PERFORM-ERA "Postdoctoral Performance for Integration in the European Research Area" (ID-57649), financed by the European Social Fund and the Romanian Government.

References

1. Shor, P.: Algorithms for quantum computation: discrete logarithms and factoring. In: Proc. of the 35th Annual Symp. on Foundations of Computer Science, pp. 124–134. IEEE Computer Society (1994)
2. Grover, L.: A fast quantum mechanical algorithm for database search. In: Proc. of 28th ACM Annual STOC, pp. 212–219 (1996)
3. Venegas-Andraca, S., Ball, J.: Storing images in engtangled quantum systems. arXiv:quant-ph/0402085 (2003)
4. Venegas-Andraca, S., Bose, S.: Storing, processing and retrieving an image using quantum mechanics. In: Proc. of the SPIE Conf. Quantum Information and Computation, pp. 137–147 (2003)
5. Latorre, J.: Image compression and entanglement. arXiv:quant-ph/0510031 (2005)
6. Le, P., Dong, F., Hirota, K.: A flexible representation of quantum images for polynomial preparation, image compression, and processing operations. Quantum Inf. Process. 10, 63–84 (2011)
7. Venegas-Andraca, S., Ball, J.: Processing images in entangled quantum systems. Quantum Inf. Process. 9, 1–11 (2010)
8. Fijany, A., Williams, C.: Quantum wavelet transform: fast algorithm and complete circuits. arXiv:quant-ph/9809004 (1998)
9. Klappenecker, A., Roetteler, M.: Discrete cosine transforms on quantum computers. In: IEEER8-EURASIP Symp. on Image and Signal Processing and Analysis (ISPA 2001), Pula, Croatia, pp. 464–468 (2001)
10. Lomont, C.: Quantum convolution and quantum correlation algorithms are physically impossible. arXiv:quant-ph/0309070 (2003)
11. Bruß, D., Macchiavello, C.: Optimal eavesdropping in cryptography with three-dimensional quantum states. Phys. Rev. Lett. 88, 127901 (2002)
12. Durt, T., Cerf, N.J., Gisin, N., Żukowski, M.: Security of quantum key distribution with entangled qutrits. Phys. Rev. A 67, 012311 (2003)
13. Spekkens, R.W., Rudolph, T.: Degrees of concealment and bindingness in quantum bit commitment protocols. Phys. Rev. A 65, 012310 (2001)
14. Greentree, A.D., Schirmer, S.G., Green, F., Hollenberg, L.C.L., Hamilton, A.R., Clark, R.G.: Maximizing the hilbert space for a finite number of distinguishable quantum states. Phys. Rev. Lett. 92, 097901 (2004)
15. Ralph, T.C., Resch, K.J., Gilchrist, A.: Efficient toffoli gates using qudits. Phys. Rev. A 75, 022313 (2007)
16. Lanyon, B., Barbieri, M., Almeida, M., Jennewein, T., Ralph, T., Resch, K., Pryde, G., O'Brien, J., Gilchrist, A., White, A.: Simplifying quantum logic using higher-dimensional hilbert spaces. Nature Physics 5(2), 134–140 (2008)
17. Collins, D., Gisin, N., Linden, N., Massar, S., Popescu, S.: Bell inequalities for arbitrarily high-dimensional systems. Phys. Rev. Lett. 88, 040404 (2002)
18. Acin, A., Durt, T., Gisin, N., Latorre, J.I.: Quantum nonlocality in two three-level systems. Phys. Rev. A 65, 052325 (2002)
19. Vallone, G., Pomarico, E., De Martini, F., Mataloni, P., Barbieri, M.: Experimental realization of polarization qutrits from nonmaximally entangled states. Phys. Rev. A 76, 012319 (2007)
20. Neeley, M., Ansmann, M., Bialczak, R.C., Hofheinz, M., Lucero, E., O'Connell, A.D., Sank, D., Wang, H., Wenner, J., Cleland, A.N., Geller, M.R., Martinis, J.M.: Emulation of a Quantum Spin with a Superconducting Phase Qudit. Science 325, 722 (2009)

21. Straupe, S., Kulik, S.: Quantum optics: The quest for higher dimensionality. Nature Photonics 4, 585–586 (2010)
22. Bianchetti, R., Filipp, S., Baur, M., Fink, J.M., Lang, C., Steffen, L., Boissonneault, M., Blais, A., Wallraff, A.: Control and tomography of a three level superconducting artificial atom. Phys. Rev. Lett. 105, 223601 (2010)
23. Muthukrishnan, A., Stroud, C.R.: Multivalued logic gates for quantum computation. Phys. Rev. A 62(5), 052309 (2000)
24. Perkowski, M., Al-Rabadi, A., Kerntopf, P.: Multiple-valued quantum logic synthesis. In: Proc. of Intl. Symp. on New Paradigm VLSI Computing, Sendai, Japan, December 12-14, pp. 41–47 (2002)
25. Miller, D., Maslov, D., Dueck, G.: Synthesis of quantum multiple-valued circuits. J. Multiple-Valued Logic Soft Comput. 12(5-6), 431–450 (2006)
26. Khan, M., Perkowski, M.: Genetic algorithm based synthesis of multi-output ternary functions using quantum cascade of generalized ternary gates. In: Congress on Evolutionary Computation, vol. 2, pp. 2194–2201 (2004)
27. Mohammadi, M., Niknafs, A., Eshghi, M.: Controlled gates for multi-level quantum computation. Quantum Inf. Process. 10(2), 241–256 (2011)
28. Yang, G., Song, X., Perkowski, M.A., Wu, J.: Realizing ternary quantum switching networks without ancilla bits. Journal of Physics A: Mathematical and General 38(44), 9689–9697 (2005)
29. Mandal, S., Chakrabarti, A., Sur-Kolay, S.: Synthesis techniques for ternary quantum logic. In: 41st IEEE Intl. Symp. on Multiple-Valued Logic, pp. 218–223 (2011)
30. Giesecke, N., Kim, D.H., Hossain, S., Perkowski, M.: Search for universal ternary quantum gate sets with exact minimum costs. In: Proc. of RM Symp., Oslo, May 16 (2007)
31. Zadeh, R.P., Haghparast, M.: A new reversible/quantum ternary comparator. Australian Journal of Basic and Applied Sciences 5(12), 2348–2355 (2011)
32. Khan, M.H.A.: Design of reversible/quantum ternary comparator circuits. Engineering Letters 16(2), 178–184 (2008)
33. Brassard, G., Hoyer, P., Tapp, A.: Quantum counting, arXiv:quant-ph/9805082v1 (1998)
34. Williams, C.: Explorations in Quantum Computing, 2nd edn. Springer (2011)
35. Klimov, A.B., Guzmán, R., Retamal, J.C., Saavedra, C.: Qutrit quantum computer with trapped ions. Phys. Rev. A 67, 62313 (2003)
36. Zilic, Z., Radecka, K.: Scaling and better approximating quantum fourier transform by higher radices. IEEE Trans. Comput. 56(2), 202–207 (2007)
37. Fan, Y.: Applications of multi-valued quantum algorithms (2007), http://arxiv.org/pdf/0809.0932
38. Li, H., Wu, C., Liu, W., Chen, P., Li, C.: Fast quantum search algorithm for databases of arbitrary size and its implementation in a cavity qed system. Phys. Lett. A 375(48), 4249–4254 (2011)
39. Parasa, V., Perkowski, M.: Quantum phase estimation using multivalued logic. In: IEEE Intl. Symp. on Multiple-Valued Logic, pp. 224–229 (2011)

Firefly-Inspired Synchronization of Sensor Networks with Variable Period Lengths

Stefan Wieser, Pier Luca Montessoro, and Mirko Loghi

Dept. of Electrical, Managerial and Mechanical Eng., University of Udine, Italy
swieser@itec.aau.at, {montessoro,mirko.loghi}@uniud.it

Abstract. Synchrony is an important requirement in wireless sensor networks. Biologically inspired synchronization has received significant scientific attention for its simplicity and robustness. This paper presents a self-organized approach based on two established synchronization methods, Phase-Advance and the Reachback Firefly Algorithm, that can be used if the phase and the period length of a node must both be adjusted. By considering nodes with different period lengths, we account for and synchronize nodes with variable delays and clock inaccuracies. We evaluate our modifications through extensive simulations with a realistic model.

Keywords: Biologically inspired synchronization, wireless sensor networks.

1 Introduction

Synchronization has been observed in many organisms: fireflies in South-East Asia flash in unison during mating [1], snowy tree crickets synchronize their chirps [2]. These creatures use simple cues from other individuals to adapt and synchronize themselves. A large amount of scientific work is based upon biologically inspired synchronization: they also use simple interactions, such as light impulses from other nodes, and achieve synchrony in a similar manner.

Previous work that uses bio-inspired approaches to synchronize the phase of wireless nodes assumes that their *period lengths* are either identical [3], [4], or that nodes are able to exchange some information about their period lengths [5], [6]. In the following, when we refer to periods for brevity, we are referring to the length (the duration in a certain time frame, possibly seconds) of the period.

This paper presents a novel approach that can be used if both, the phase and the period of a node must be adjusted. This allows, for example, the synchronization of nodes with not negligible clock inaccuracies. Synchrony is achieved without exchanging any information between the nodes (beyond physical layer information that a pulse has been emitted by some node). The synchronization algorithm presented in this paper achieves synchrony even if synchronization signals collide, and synchronizes nodes with different periods, different clock inaccuracies and different initial phases. Extensive simulations with a realistic model are used to evaluate our modifications.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 376–385, 2013.

2 Related Work

Biologically inspired synchronization has received significant scientific attention. Mirollo and Strogatz introduced a mathematical model, Phase-Advance (PA), showing that two oscillators, responding to other *synchronization signals* by advancing their own phase (and thus emitting their pulse earlier), will synchronize under almost all conditions [7]. This also holds for a larger population of oscillators, under the condition that the network is fully connected (all oscillators can sense each others pulses). Two synchronized oscillators that trigger at the same time are perceived as a single oscillator emitting stronger pulses. The strengths of the pulses does not need to be additive (for example, the combined pulses of two synchronized oscillators is not perceived as twice as strong by other nodes). The proof in [7] also holds for multi-hop networks that are not fully connected [8].

Both models uses several assumptions that are not realistic in wireless sensor networks [3]: specifically, they assume that emitting and observing a pulse, as well as computation, occur instantaneously. Medium access control (MAC) may cause delays that are not negligible. The Reachback Firefly Algorithm (RFA) [3] considers these latencies by making nodes only record the time they observe a pulse from another node, without adjusting their current phase. When their current phase completes, they adjust their following phase based on all the recorded pulses. Such an adjustment must still be computed based on a "smooth, monotonically increasing, and concave down" state correction function [7], but the amount of computations each node must perform is reduced to one operation. In addition, the computation occurs directly after the node emits a pulse, thus a time where its duration is not as critical. Preemptive Message Staggering is used to reduce the likelihood that all nodes transmit their synchronization pulses at the same time. A modification of their state correction function has been performed by [9], which achieves faster convergence time by indirectly clustering the pulses. It should be noted that for both approaches, this adjustment is only applied to the following phase - the "default period" of a node remains unchanged. As a result, it is necessary that all nodes have the same period $\mathcal{P}$.

Another interesting approach is the S-MAC protocol [5] that reduces the four major sources of energy waste in mobile nodes: (i) collisions (nodes transmit at the same time, rendering the received signal useless), (ii) overhearing (a node receives data targeted at another node), (iii) control overhead (management data that must be sent for medium access) and (iv) idle listening (a node listens for potential data but receives nothing).

Under the assumption that sensor networks are idle most of the time (unless the sensors detect something) and some latency is acceptable, the S-MAC protocol allows nodes to sleep for a fraction of their duty cycle. As a result, a sending node may need to repeat its transmission at a different time in order to ensure that the target node is listening. Each node periodically broadcasts its sleeping schedule. The first node to broadcast its schedule on the medium becomes the *synchronizing node*. New nodes listen to the medium for a "certain period" to receive this schedule, and become *followers*. If nodes fail to discover each other,

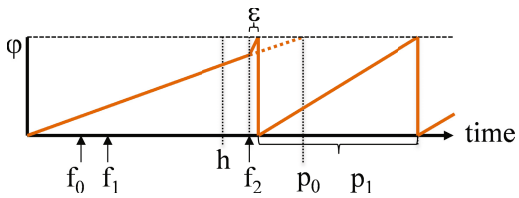


Fig. 1. Illustration of a node's phase φ, during which three flashes ($f_{0..2}$) are received

they adopt multiple schedules, reducing the energy saved. However, collision avoidance must be performed for synchronization messages, which contain data needed by other nodes, thus limiting the scalability of the approach.

3 Synchronization Algorithm

Our approach differs from the aforementioned ones in two main features: first, we do not transmit any data in our synchronization pulses and thus do not require any collision avoidance protocol. Second, and more importantly, we do not require that all nodes have the same period $\mathcal{P}$. In fact, nodes using our algorithm synchronize their phases, as well as their periods. In this section, we present our adaptations that allow nodes to gradually increase their synchronization periods, reducing the energy wasted on control overhead, even in the presence of clock inaccuracies. Based on the common notion of $\mathcal{P}$, nodes can then apply established techniques to reduce idle listening, overhearing, and collision. Our approach is separated into three stages: the core of the algorithm is an initial *synchronization stage*, followed by a *pre-data stage*, during which any node not yet synchronized is detected, and finally the *data stage*, during which nodes exchange data.

Referring to Figure 1, we assume that a node has an initial period length p_0 at which it emits synchronization signals. Based on the signals the node receives from other nodes (f_0, f_1 and f_2) during its i^{th} period (p_i), it adjusts its $i + 1^{th}$ period (p_{i+1}). No specific type of signal is required for synchronization, it is sufficient that nodes can detect its presence. In particular, no data needs to be encoded in a signal. Nodes do not exchange any information about what their periods are, which allows us to, for example, implicitly account for clock drift or other hardware delays. Furthermore, a node may "miss" signals: other than increasing the time needed to synchronize, this has no lasting effect.

3.1 Synchronization Stage

During the first stage (the *synchronization stage*), we synchronize the periods of the nodes by using an adaptation of RFA, and the phases by using an adaptation of PA.

Adapted Phase Advance Algorithm. It is suggested that applying PA to mobile nodes is difficult for two reasons [3]: PA assumes a node is able to compute

how much the current phase must be advanced instantaneously, and that it can emit a signal immediately if needed. In our adapted version these assumptions are relaxed. First, PA is only used at most once per phase: if a signal is received after a threshold h (which a node computes at the beginning of its phase), the timestamp for the RFA is recorded, and the node emits its signal. Therefore, expensive computations during the phase are not necessary. h is based on a node's current period, p, and a dampening variable, d, which is initially set to 0.6. Every time after a node emits a synchronization signal, it increases d by a constant factor c_d, adjusts its period based on RFA, and calculates the new threshold $h = d \cdot p$. Effectively, increasing d ensures that nodes rely less on only aligning the phases as time advances, and more on the modified RFA, which both aligns the phases, and adjusts the periods.

Second, we do not require that a node flashes immediately. For example, a node may have a computational delay of ϵ (see Figure 1) before it emits the signal. In our simulations, we added a random delay of up to $0.3s$ if a node rescheduled its period. Synchronization is still possible because while PA is used in conjunction with RFA to speed up synchronization, the final synchronization is achieved with RFA. One could argue that PA should not be included. However, the speed up even with this adapted version of PA is substantial. In preliminary experiments with 10 nodes, using the adapted version of PA in conjunction with RFA reduced the time spent to reach the data stage by 63.11%.

Adapted Reachback Fireflies Algorithm. A node adjusts its *period* with an adapted version of RFA[1]. Recall that RFA resets the records after every flash. Since a node that detects no other signal returns to its original period p_0, nodes with different periods cannot be synchronized with this approach.

To tackle this issue, we modified RFA to not only change the following phase, but also the period of a node. Essentially, a node using our adapted algorithm no longer returns to p_0. This permits synchronizing nodes with different periods. A naive solution is to modify RFA so that the phase change not only affects a node's current phase, but also all of its following phases. However, this causes nodes to synchronize at a very short period (the exact period is then determined by other factors such as propagation delay, recharge time and the duration of the synchronization signals), which are not desirable due to energy concerns and the overhead of keeping nodes synchronized.

One reason that periods can become this short is that PA and RFA only decrease the duration of a phase, and never increase it. The obvious approach to avoid this, namely to also increase its length, is not an option, because in that case the synchronization of the system is no longer guaranteed [7].

Instead, we follow a different approach: in order to ensure that periods do not become too short, we apply a dampening factor to the adjustment a calculated by

[1] A node following the original RFA records the time of every synchronization signal it receives, without shortening its phase. Only after the node has emitted its own signal, it reduces the time of its next phase, based on its records of the signals it received. The time each signal was received affects how much the next phase is reduced by; signals received later in the phase result in a larger reduction.

RFA. This allows nodes with very different periods to align themselves quickly, without causing nodes with already short periods to decrease them even more. A node uses this dampening factor to modify its i^{th} period p_i at the end of each phase, as shown in (1).

$$p_{i+1} = p_i - \frac{a \cdot (wp_i)^2}{C} \tag{1}$$

Several factors are used to weight the aforementioned adjustment a. This is done to prevent periods from becoming too short. w is a weight from 0 to 1 that indicates how close to p the last period was. For example, in Figure 1 f_2 triggers PA and ends its current period, therefore $w = f_2/p_0$. If PA is used to advance the phase, w decreases, and period adjustment is slowed. This helps synchronizing a system where the periods are almost equal, but the phases are not yet aligned. C is a constant that influences how quickly the system synchronizes, and also the final period the nodes synchronize at. When C is increased, the time required until the system stabilizes and the final period $\mathcal{P}$ both increase as well. We found that for our simulations, setting $C = 100$ adjusts a node's period quickly, resulting in fast synchronization times, without lowering the final period $\mathcal{P}$ too much. Finally, a node determines the new adjusted period p_{i+1} by subtracting this weighted adjustment from its current period p_i.

Another change in our approach is the lack of preemptive message staggering. RFA transmits a timestamp in the synchronization signal to estimate the delay of the medium. In addition to requiring synchronized clocks, situations in which multiple synchronized nodes emit signals at the same time must also be avoided, as a collision would render the transmitted data useless. RFA uses preemptive message staggering to avoid collisions. We found that explicitly considering the medium delay was not necessary because adjusting the phases and the periods already consider it implicitly. We do not need to transmit any data in the signals; the *presence* of a signal is all the information we require. As a result, collisions have no detrimental effect on our algorithm and neither collision avoidance, nor collision detection, nor preemptive message staggering are needed.

Essentially, a node follows the steps outlined in Algorithm 1.

Algorithm 1. Synchronization Stage. t represents the current time.

1: At the beginning of phase:
2: $p \leftarrow p - a \cdot (wp_i)^2/C$ ▷ Adjust p based on previous phase
3: $d \leftarrow d + c_d$, $h \leftarrow d \cdot p$, $t_s \leftarrow t$ ▷ Set threshold and record beginning of phase
4: If a signal is received during the phase, before the end of the period:
5: $a \leftarrow a + 0.08 \cdot (t - t_s)$ ▷ PA (a linear function is used for simplicity)
6: **if** $t \geq t_s + h$: emit signal and end phase

3.2 The Pre-data Stage

In order to determine when to switch to the pre-data stage, we distinguish between nodes that are *leaders* and *followers*. A node considers itself a leader if it detected no signals from other nodes during its last phase before it emits its own signal; otherwise it considers itself a follower. Note that it is possible several nodes consider themselves leaders at the same time, if they emit their signals near-simultaneously, so that propagation and processing delays prevent a node from receiving signals from other nodes before emitting its own. This effect is desirable for our algorithm, as it prevents simultaneous signals of other synchronized nodes from affecting the node's following phase, and occurs frequently once nodes are synchronized. Similar to MEMFIS [10], nodes in our simulation use a refractory period of $12\mu s$ after they finish emitting their signals, during which they do not consider signals from other nodes.

When a node determines that is a leader, it immediately switches to the pre-data stage. During the pre-data stage, it increases the length of its period by a small amount of time, Δ, every time it emits a synchronization signal. If it receives any other synchronization signal outside the refractory period, it switches back to the synchronization stage.

Increasing the period serves three important purposes: first, it allows a node to detect other nodes that are not in the same stage. A node that is still in the synchronization stage will either keep the same period (if it receives no other signals), or decrease its period, but never increase it. If it detects nodes that are not yet synchronized, the node switches back to the synchronization stage. Second, a longer period between synchronization messages allows a node to save energy, as it does not need to wake up frequently. We can make periods as long as desired by increasing the time that nodes spend in the pre-data stage. Third, note that RFA and PA both only reduce the phase of a node, and therefore the periods in the synchronization stage get shorter, not longer. The pre-data stage allows such nodes to synchronize, without reducing the periods of other existing nodes. This is particularly important for late joining nodes, discussed later, which are not permitted to flash until they are synchronized. If a node joins that has a shorter period than any other node in the system, it will determine that it is the leader, switch to the pre-data stage, and begin increasing its period.

Assume, for now, that all nodes are synchronized, but still in the synchronization stage. Due to the propagation and processing delay, and the refractory period after emitting a signal, none of the nodes will detect a synchronization signal from any other node. As a result, all nodes consider themselves leaders, do not change their period, and therefore switch to the data stage. During the pre-data phase, all nodes begin increasing the length of their period, again in perfect synchronization.

Consider the case where a one node is not yet synchronized (for example, a node that arrives late, or a node that missed signals and thus considers itself synchronized). This node detects the signal from the synchronized nodes increasing the length of their periods, and uses RFA to shorten its own. This is, in turn, detected by the already synchronized nodes, which return to their

synchronization stage to synchronize with the aforementioned node. It should be noted that gradually increasing the periods is not equivalent to simply setting the periods of all nodes to a fixed value (such as three seconds). Our adaptations ensure that the *actual* periods are equal, even if individual nodes have a different understanding of time. This can occur, for example, if a node's clock is fast, or slow, or because it experiences a waking up delay that is different from the delay experienced by other nodes. Our simulations show that a node experiencing a $0.5s$ delay will synchronize at $\mathcal{P} = 2.5s$ if other nodes have a period of $\mathcal{P} = 3.0s$. This effect is achieved without explicitly considering such delay.

3.3 The Data Stage

After a node has remained in the pre-data stage for a predefined number of periods, it switches to the data stage. In the data stage, a node only considers signals for synchronization that are received in a short-time span δ before and after it emits its own synchronization signals. For the remaining time, the medium is used for transmitting data. We do not require a specific MAC protocol; any protocol (such as TDMA or CDMA) may be used, provided that it is modified to not transmit during $\mathcal{P} \pm \delta$. Note that it is necessary that nodes consider propagation delay and the refractory period. Therefore, nodes must stop transmitting slightly earlier than δ before the next signal is due.

Late Joining Nodes. Nodes that join the network must first sense the medium for a predefined amount of time (the "initial observation period"), which must be no shorter than the period of the slowest node. In case it detects any data transmissions between the synchronization signals, it determines that it is a late joining node. In this case, it follows the same algorithm described above, with the exception of that it does not flash until it has determined itself to be leader for two subsequent periods, for a duration no shorter than the initial observation period. This simply prevents the late joining node from interrupting the data transmission of the existing, synchronized nodes.

4 Evaluation

We evaluated our modifications through simulations, using varying numbers of nodes with different phases and periods. In our model, nodes are placed in random locations in a three-dimensional space. A node can only detect a limited number of concurrent signals. In order to assume the worst case for our evaluations, we set the number of concurrent signals a node can detect to one. Therefore, if a node senses a signal, it will not sense signals from any other node, until the first node stops emitting the signal. Furthermore, we assume that nodes are not able to distinguish between signal strengths. These restrictions make it more difficult for a node to synchronize, as they are no longer able to distinguish between a stray, unsynchronized node, and a group of already synchronized nodes. Finally, the model includes a probability nodes may miss signals entirely,

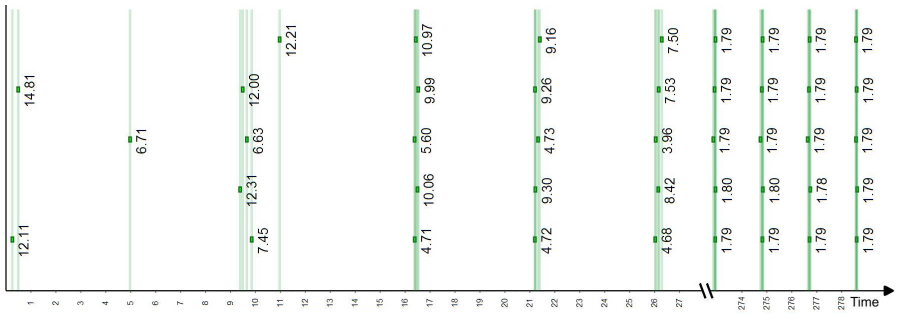


Fig. 2. Synchronization of the phases and periods of a 5 node system

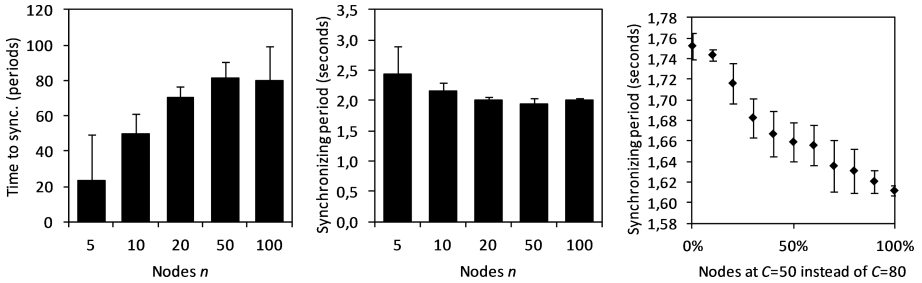
based on the distance between two nodes, and considers propagation delay of the MAC, and processing delay of the nodes. For our simulations, nodes are placed in a $10^6 m^3$ air-filled cube, and use electromagnetic waves to communicate. We measured the time required to reach the data stage, and the final period nodes synchronize at. Table 1 shows that the initial periods are vastly different: for example, node 8 initially emits a signal every $5.7s$, whereas node 2 uses a period of $11.0s$. The difference in periods used during our experiments is far greater than what is reasonable for actual hardware ($\pm 0.1s$ [9]). However, it serves to prove the robustness of our approach.

Table 1. Phase offset and periods (in seconds) of a 10 node simulation

Node	1	2	3	4	5	6	7	8	9	10
Offset	7.0281	4.0556	1.1924	3.0100	9.2154	6.5981	8.5397	9.1894	6.5822	5.0822
Period	9.2	11.0	6.8	4.2	10.2	9.2	9.4	5.7	8.1	9.4

We performed Monte-Carlo simulations with 5, 10, 20, 50 and 100 nodes, which are system sizes comparable to what is used in recent related work [9]. The nominal period of every node is set to $8.0s$, with a timing error of $\pm 50\%$. The synchronization signal is a $0.1s$ electro-magnetic pulse, and a node switches to the data stage if it remained in the pre-data stage for ten periods. Each simulation was repeated several times to obtain some statistical significance.

In our simulations, nodes that attempt to flash immediately are subjected a random delay of up to $0.3s$ to account for delays in hard- and software. This is illustrated in Figure 2, which shows the synchronization of five nodes over time. Squares in the graph represent flashes. Light lines are associated which each flash, showing when the nodes flash (in respect to the other nodes). The number on each flash indicates the node's current period. The actual period can be shorter due to PA, if a signal is received after the threshold h. For example, consider the bottommost node. The node initially has a period of $p_0 = 12.11$. It receives two signals (after 0.2 and 5 seconds), however since they were received



(a) Time to synchronization

(b) Final periods $\mathcal{P}$

(c) Final period $\mathcal{P}$ for varying C

Fig. 3. Experimental results with varied number of nodes and dampening constant C

before the threshold h (not shown in the graph), it only records their times for RFA. The next signal it receives, after the 9 second mark, is received after the threshold h. Therefore, it flashes immediately. Due to the aforementioned delay ϵ, its signal is only emitted near the 10 second mark. Based on the signals received it then calculates its new period $p_1 = 7.45$. Notice that the initial periods of the nodes are very different ($14.81s$ versus $6.71s$), and that the random delay introduced for PA prevents nodes from flashing at the same time. Still, the primitive implementation of PA helps synchronizing the phases. In the following periods, nodes synchronize their period lengths, and rely on PA less.

Figure 3a displays the number of periods needed to reach the data stage when varying the numbers of nodes. The bars correspond to the 50th percentile of the simulations, whereas the error bars represent the 90th percentile. Figure 3b illustrates the final period $\mathcal{P}$ that nodes synchronize at upon reaching the data stage. Most importantly, note that the number of periods needed for synchronization increases with the number of nodes. This is a contrast to other synchronization methods, such as [9], in which the number of periods remain comparable, or even decrease with the addition of nodes. The reason for the increase in our approach is that we also synchronizes the period length, which becomes more difficult with an increasing number of nodes.

Another observation is that the final periods $\mathcal{P}$ decrease the longer nodes take to synchronize. This is expected, as both PA and RFA only decrease the length of a phase, and do not increase it. However, the decreasing in period length remains reasonable, thanks to the dampening factor C, introduced in Section 3.1.

It should be noted that in our experiments, we set $C = 100$ on all nodes. However, it is possible that nodes use different values of C (think, for example, a factor introduced by clock inaccuracies). Therefore, we performed a separate set of experiments with 10 nodes that use initial periods of $8.0s \pm 50\%$. We set a percentage of nodes to $C = 50$, while the remaining nodes use $C = 80$. Figure 3c shows the average final period $\mathcal{P}$, if the percentage of nodes using $C = 50$ changes. The error bars represent the 95% confidence interval. Note that the final period, which nodes synchronize at, gradually decreases as the percentage of nodes using the $C = 50$ increases. This demonstrates that the

algorithm is not only able to synchronize nodes with different periods, but also nodes with different timing, and that the resulting periods in this situation are reasonable.

5 Conclusion and Future Work

In this paper we presented an adaptation of RFA and PA that synchronizes nodes where both phase and period are different. We showed that synchronization with large numbers of nodes with periods that differ by over 100% is feasible. A notable observation is that in the experiments where half of the nodes used the dampening constant $C = 50$ and the other half used $C = 80$, the resulting final period was close to the average of what we measured when all nodes used the same C (i.e. $C = 50$ or $C = 80$). An interesting extension is exploring how C, the nodes' initial periods and the adjustment from RFA interact. However, this is subject of ongoing research.

References

1. Buck, J.: Synchronous rhythmic flashing of fireflies. II. The Quarterly Review of Biology, 265–289 (September 1988)
2. Walker, T.J.: Acoustic synchrony: Two mechanisms in the snowy tree cricket. Science 166, 891–894 (1969)
3. Werner-Allen, G., Tewari, G., Patel, A., Welsh, M., Nagpal, R.: Firefly-inspired sensor network synchronicity with realistic radio effects. In: Proc. 3rd Int. Conference on Embedded Networked Sensor Systems, SenSys 2005, pp. 142–153. ACM, New York (2005)
4. Rentel, C., Kunz, T.: A clock-sampling mutual network time-synchronization algorithm for wireless ad hoc networks. In: 2005 IEEE Wireless Communications and Networking Conference, vol. 1, pp. 638–644 (March 2005)
5. Ye, W., Heidemann, J., Estrin, D.: An energy-efficient mac protocol for wireless sensor networks. In: Proc. 21st Annual Joint Conference of the IEEE Computer and Communications Societies, vol. 3, pp. 1567–1576 (2002)
6. Yadav, P., McCann, J.A.: EBS: decentralised slot synchronisation for broadcast messaging for low-power wireless embedded systems. In: Proc. 5th Int. Conference on Communication System Software and Middleware, COMSWARE 2011, pp. 9:1–9:6. ACM, New York (2011)
7. Mirollo, R.E., Strogatz, S.H.: Synchronization of pulse-coupled biological oscillators. SIAM J. on Applied Mathematics 50(6), 1645–1662 (1990)
8. Lucarelli, D., Wang, I.-J.: Decentralized synchronization protocols with nearest neighbor communication. In: Proc. 2nd Int. Conference on Embedded Networked Sensor Systems, SenSys 2004, pp. 62–68. ACM, New York (2004)
9. Leidenfrost, R., Elmenreich, W.: Firefly clock synchronization in an 802.15.4 wireless network. EURASIP J. on Embedded Systems, 7:1–7:17 (January 2009)
10. Tyrrell, A., Auer, G., Bettstetter, C.: Emergent slot synchronization in wireless networks. IEEE Transactions on Mobile Computing 9(5), 719–732 (2010)

Phase Transitions in Fermionic Networks

Marco Alberto Javarone and Giuliano Armano

DIEE - Dept. of Electrical and Electronic Engineering
University of Cagliari, Cagliari 09123, Italy
{marco.javarone,armano}@diee.unica.it

Abstract. We show that the emergence of different structures in complex networks can be represented in terms of a phase transition for quantum gases. In particular, we propose a model of fermionic networks that allows to investigate the network evolution and its dependence on the system temperature. Simulations, performed in accordance with the cited model, illustrate that the transition from classical random networks to scale-free networks mimics a cooling process in quantum gases. Furthermore, we found that, at very low temperatures, a winner-takes-all structure emerges. We deem this model useful for studying the evolution of complex networks and also for representing competitive dynamics.

Keywords: complex networks, quantum statistics, network evolution, competitive dynamics.

1 Introduction

Many natural and man-made complex systems can be modeled by complex networks, i.e., biological cells and their interactions, social dynamics, world wide web, and every other system containing a notable number of interacting elements ([1][2][3]). In general, complex networks allow to represent systems at a non-equilibrium state, meaning that new nodes and/or new links can be added over time during an evolutionary, often irreversible, process. In [1], Albert and Barabasi illustrated the central role of statistical mechanics for analysing these networks. One of the first models for random graphs dynamics has been developed by Erdos and Renyi, see [4]. This kind of networks, called *ER graphs*, has low clustering coefficient and a binomial degree distribution of nodes, converging to a Poissonian distribution for large number of nodes. Later on, in their *WS model*, Watts and Strogatz [5] interpolated random networks with a regular ring lattice, achieving networks characterized by short average path lengths, high clustering coefficient, and homogeneous degree distribution of nodes. Observing that in many real networks the degree distribution of nodes follows a power-law, Barabasi and Albert developed the *BA model* [1], defining the concept of *scale-free* networks. The power-law equation is the following:

$$P(k) \sim c \cdot k^{-\gamma} \tag{1}$$

where k is the node degree, c is a normalizing constant and γ is a parameter, known as scaling parameter, usually in the range $(2, 3)$. A notable consequence is

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 386–395, 2013.

that, in these networks, only few nodes (called hubs) have a big number of links. The evolutionary process of networks has been studied by many authors, often resorting to theoretical physics. For the sake of brevity, let us cite only few. Bianconi and Barabasi [6] compared Bose-Einstein Condensation phenomena (BEC) to winner-takes-all policies; Bianconi [7] discussed the symmetric construction of bosonic and fermionic networks, asserting that the former networks are scale-free and the latter are growing Cayley trees. Kriukov et al. [8] developed a geometric framework to study the structure and functions of complex networks, interpreting edges as non-interacting fermions whose energies are hyperbolic distances between nodes. Shen Yi et al. [9] discussed an inverse approach to network evolution defining a relation with their model and Fermi-Dirac statistics. Baronchelli et al. [10] defined a framework using bosonic reaction-diffusion processes, with the aim of analysing dynamical systems on complex networks. Perseguers et al. [11] developed a model of quantum complex networks, drawing a link between the field of complex networks and that of quantum computing.

In this work we propose a theoretical model of network evolution based on Fermi-Dirac statistics, obtained by mapping complex networks to fermionic gases. The proposed model, called fermionic networks, which can be seen as dual with respect to bosonic networks ([6]), shows that the emergence of different structures can be represented in terms of a phase transition for quantum gases. In particular, at low temperatures networks are winner-takes-all, at intermediate temperatures are scale-free, whereas at high temperatures they become classical random networks. We consider fermionic networks useful also to represent competitive dynamics since, at low temperatures, few winners nodes arise. The remainder of the paper is organized as follows: Section 2 gives a brief introduction to quantum statistics. Section 3 describes some existing models of networks as non-interacting particle systems. Section 4 introduces the proposed model of network dynamics, based on Fermi-Dirac statistics. Section 5 shows the results of the corresponding simulations. Conclusions (i.e., Section 6) end the paper.

2 Quantum Statistics

Statistical mechanics assumes a central role when dealing with systems composed by many particles, the underlying assumption being that particles are identical and indistinguishable. Moreover, their quantum energy levels are extremely closely spaced, with a cardinality much greater then the number of particles. Energy levels can be grouped in bundles with the approximation that levels in the same bundle have the same energy. Particles with a symmetric wave function, called bosons, obey Bose-Einstein statistics, whereas particles with an antisymmetric wave function, called fermions, obey to Fermi-Dirac statistics –see [12]. Given a system with N particles of the same type, we can build an N-body wave function, with several admissible states. For each state α, the corresponding number of particles, say n_α (also called occupation number), is given by the following equation:

$$n_\alpha = \begin{cases} 0, 1, ..., \infty & \text{bosons} \\ 0, 1 & \text{fermions} \end{cases} \tag{2}$$

and $\sum_\alpha n_\alpha = N$. In the event that a gas is composed by fermions, we must consider also the Pauli exclusion principle. Overall, the number of microstates in a macrostate is computed by the equation:

$$\Omega_f = \Pi_i \frac{g_i!}{n_i!(g_i - n_i)!} \tag{3}$$

with g_i representing the i-th bundle. The distribution of particles follows the Fermi-Dirac statistics:

$$n_i^f = \frac{g_i}{e^{\frac{\epsilon_i - \mu}{k_b T}} + 1} \tag{4}$$

where ϵ_i denotes the energy of the i-th bundle, μ the chemical potential, and k_b the Boltzmann constant.

This distribution approaches the classical behaviour in proximity of the high-temperature limit, showing a quantum-classical transition. This phenomenon occurs when particles occupy excited states sparsely. In particular, considering the thermal wavelength λ and the density ρ, the following conditions can be stated:

$$\begin{cases} \rho\lambda^3 \gg 1 & \text{classical regime} \\ \rho\lambda^3 \approx 1 & \text{onset of quantum effects} \end{cases}$$

The classical regime is described by the Maxwell-Boltzmann distribution. In particular, with $Z = \sum_j g_j e^{-\frac{\epsilon_j}{k_b T}}$ partition function, we write:

$$n_i^{mb} = \frac{N}{Z} g_i e^{-\frac{\epsilon_i}{k_b T}} \tag{5}$$

3 Networks as Particle Systems

Under certain assumptions, complex networks can be considered as thermodynamic systems that evolve from one state to another [13]. In this section we give a synthetic overview of some existing models of network dynamics based on quantum statistics.

3.1 Bosonic Networks

In this model, Bianconi and Barabasi [6] compared network evolution to phase transition of bosonic gases. Two main structures, i.e., *fit-get-rich* and *winner-takes-all*, are identified as two different phases at low temperatures. In this model, each node is interpreted as an energy level and each link as a pair of particles.

Starting from a fitness parameter η, energy is computed according to the following equation:

$$\epsilon = -\frac{1}{\beta}\log\eta \tag{6}$$

with $\beta = \frac{1}{T}$. Here, the fitness parameter η describes the ability of each node to compete for new links. In particular, for the i-th node, the probability of connection with new nodes is proportional to:

$$\Pi_i = \frac{\eta_i k_i}{\sum_j \eta_j k_j} \tag{7}$$

with k_i degree of the i-th node. Notably, new nodes tend to link with pre-existing nodes having high values of (η, k). The generation of a scale-free network in the *fit-get-rich* phase is characterized by Equation (1) and entails the presence of few nodes with a high degree connected to many others with low degree. In a bosonic gas, when the temperature decreases, particles aim to occupy lower energy levels. Then, at a temperature below the critical temperature, the Bose-Einstein condensation takes place. In this model, as T decreases, many particles move to lower levels while keeping the corresponding particles at upper levels. In doing so, links concentrate on few nodes, until they condensate in the *winner-takes-all* phase, characterized by the fact that only one node predominates. In [7], Bianconi discussed the differences between bosonic and fermionic networks, showing that the former are scale-free, whereas the latter can be represented by Cayley trees.

3.2 Fermi-Dirac Statistics of Complex Networks

In this work [9], the authors proposed an inverse approach to network evolution, starting from a random network with an average degree equal to $2m$. Using an *illness model*, at each step, one randomly-selected node becomes "ill", then its illness causes a loss of links. An illness parameter I is defined and assigned randomly to each node. The probability for each node to lose a link depends on I and on the node degree. Hence, the number of links decreases with time. Each node has an energy defined by:

$$\epsilon = -\frac{1}{\beta}\log I \tag{8}$$

with $\beta = \frac{1}{k_b T}$. A link between two nodes corresponds to a couple of non-interacting particles placed in two energy levels whose value is computed by Equation 8. When a new node joins the network, a new energy level is added, with m new particles, and other m links are deleted, due to the illness. Authors showed that the behavior of such system can be approximated by a Fermi distribution, so that the network can be seen as a Fermi gas.

4 Fermionic Networks

Let us now introduce a novel proposal for modeling network dynamics, inspired from the physics of fermions. Given a network $G = (V, E)$, with V non empty set of nodes and E non empty set of links, let us represent each link as a particle and each node as a degenerate bundle of energy levels. Usually, the number g_i of available states in the i-th bundle is much bigger than the number p_i of its particles. Let us assume that the i-th bundle has an energy ϵ_i. This value can be assigned randomly or depends on a property of the system –e.g. a fitness parameter η or any other function deemed relevant, with the trivial constraint that it must be computable for each node of the network. In the proposed model, lower bundles have more energy levels. In particular, the first bundle has $n - 1$ levels, the second has $n - 2$ levels, and so on and so forth. Note that the link l_{ij}, which connects nodes i and j, is represented only by a single energy level, i.e., ϵ_{ij}, which in turn belongs to the i-th bundle (under the assumption that the i-th bundle is deeper than the j-th). In so doing, the last node, say y_0, is represented by a bundle without energy levels, although it can be linked in the event that a particle stays at the ϵ_{xy_0} level, with x corresponding to one of the other nodes.

4.1 Modeling Network Evolution

Let us consider an evolving network, i.e., a network that changes over time. Almost all real networks evolve over time; examples are social networks (where people find or lose friends or co-workers) and the web (where web-sites compete to gain more inlinks). Furthermore, let us consider this network a closed system, so that the number of nodes and the number of links remain constant over time. As discussed before, for every node, a bundle is defined –whose energy is computed with Eq. (6). In so doing, the relative position of each bundle depends on the value of its energy, so that deeper bundles embody more states. Considering the ability of the particles to jump between energy levels as the temperature varies, at high temperatures particles follow the classical Maxwell-Boltzmann distribution, being spread among the available states according to Eq. (5). On the other hand, as temperature decreases, many particles move to lower energy levels (see Figure 1). During a cooling process, few nodes gain new links and their degree k_i is increased. Hence, as temperature decreases, and assuming that the number of particles approximates the number of bundles, the *WTA* phase takes place (see also [6]). For every variation of the temperature, the probability for a particle to jump from the i-th to the j-th bundle is computed according to the following equation:

$$p(i \to j) = \frac{\Delta T}{T} \cdot \frac{1}{\Delta B(j,i)} \cdot f(g_j) \tag{9}$$

where T denotes the temperature of the system before the variation, ΔT the variation of temperature, $\Delta B(j, i)$ is the distance between the bundles j and i, and $f(g_j)$ is the function:

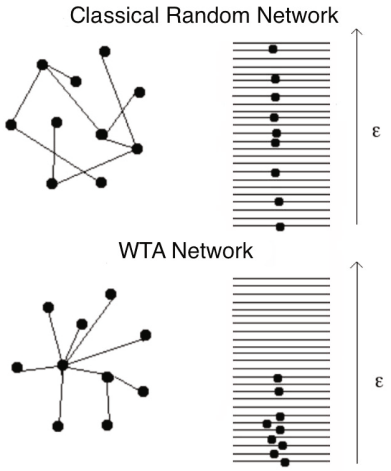


Fig. 1. On the left, from top to bottom, the evolution of a network with 10 nodes and 9 links from a classical random network to a *WTA* network. On the right, their corresponding fermionic models, which result from a cooling process that pushes particles to low energy levels.

$$f(g_j) = \begin{cases} 0 & \text{if } g_j = 0 \\ 1 & \text{if } g_j \geq 1 \end{cases} \tag{10}$$

with g_j number of available states in the j-th bundle. Hence, considering that a particle in the i-th bundle can jump to $i-1$ underlying bundles, each with a probability given by Eq. (9), the probability p_J to jump from the i-th to another bundle is:

$$p_J(i) = \sum_{z=1}^{i-1} p(i \to z) \tag{11}$$

and the probability p_S to stay in the same bundle is

$$p_S(i) = 1 - p_J(i) \tag{12}$$

Then, the final bundle of each particle is chosen by a weighted random selection among all candidate bundles (including the bundle in which the particle is located). In our model, the temperature corresponds to a phenomenon that leads to an evolution (e.g., the evolution of relations among people or a competition for new inlinks among web-sites). To complete the model, let us assume that each network has the structure of an E-R graph when generated at time $t = 0$.

5 Results

The proposed fermionic model has been tested with many simulations. In particular, we generated networks of different sizes with an Erdos-Renyi graph

structure. These networks are implemented by connecting nodes randomly – giving rise to a graph $G(n, \zeta)$, where n is the number of nodes and ζ is the probability of an edge to be drawn (note that an edge is drawn independently from other edges). Their degree distribution of nodes is binomial, converging to a poissonian distribution for a large number of nodes and small values of ζ, according to the following definition:

$$P(k) \sim e^{-\zeta n} \cdot \frac{(\zeta n)^k}{k!}$$

To generate these networks we used the following simple algorithm:

1. Define the number of n of nodes and the probability ζ for each edge
2. Draw each edge with probability ζ

Simulations have been performed with a number of nodes ranging from 100 to 5000, $\zeta = \frac{\langle k \rangle}{n-1}$ with $\langle k \rangle$ average degree of the network (see [14]) and an initial temperature ranging from 100K to 500K. For each simulation the network evolved for 50 time steps, decreasing the temperature at each step of the 10% and computing new positions of the particles. Figure 2 illustrates the evolution of the degree distribution of a network with 5000 nodes and $\langle k \rangle = 20$, generated at 100K. The normalizing costant c and the scaling parameter γ are computed at each time step. The former is computed by the following equation:

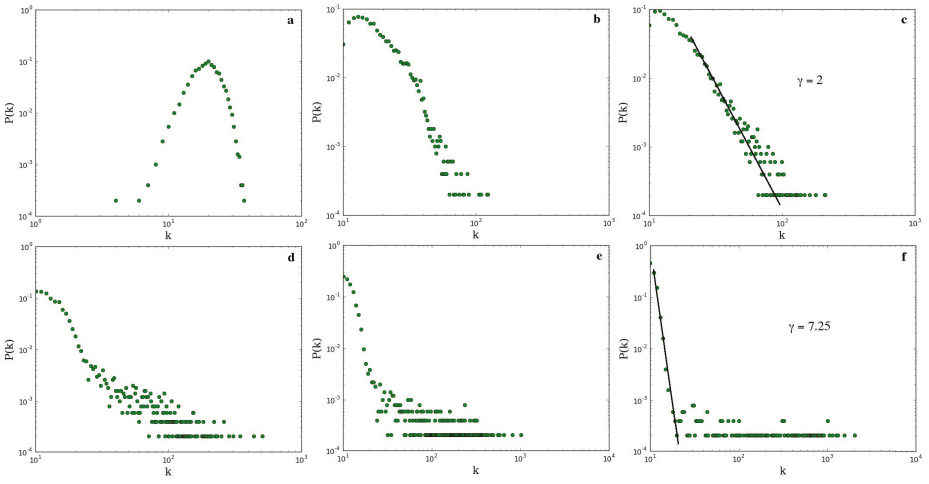


Fig. 2. On the left, from top to bottom, the evolution of the degree distribution of a network with 5000 nodes and $\langle k \rangle = 20$, generated at 100K. **a** the network at $t = 0$, it has a *ER* graph structure. **b** the network at $t = 3$. **c** the network at $t = 5$, it has a scale-free structure with a scaling parameter $\gamma \geq 2$. **d** the network at $t = 10$. **e** the network at $t = 20$. **f** the network at $t = 43$, it has a *WTA* structure with a scaling parameter $\gamma \geq 7.25$. Continuous black lines are used to highlight data interpolation.

$$c = \frac{1}{\int_{k_m}^{\infty} k^{-\gamma} dk} \tag{13}$$

where k_m is the minimum degree estimated. The scaling parameters have been estimated, as suggested in [15], by the following equation:

$$\hat{\gamma} = 1 + n \cdot \left[\sum_{i=1}^{n} \ln \frac{k_i}{k_m} \right]^{-1} \tag{14}$$

After few time steps (see Figure 2 - rectangle B), the degree distribution changes drastically, converging to that of a scale-free. Similar results have been achieved also for networks with a different number of nodes and/or generated at different initial temperatures. Figure 3 shows the number of particles that, at each time step, change their energy level. As temperature decreases, particles move to lower energy levels until they occupy the deeper bundle, hence the number of particles that change their position falls to zero.

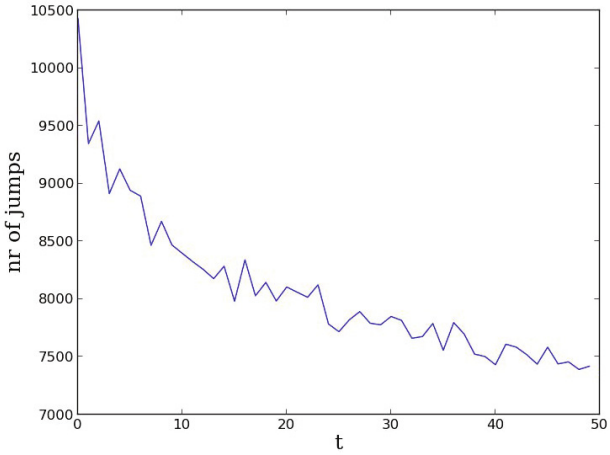


Fig. 3. Number of particles that change their energy level (indicated as nr of jumps) along time, whereas the system temperature decreases

5.1 Discussion

Fermionic networks show that the emergence of different structures can be represented as a phase transition for quantum gases. In particular, a *WTA* structure corresponds to a fermionic gas at low temperatures. On the other hand, a classical random network corresponds to the same gas at high temperatures. During a cooling process, at intermediate temperatures, a scale-free structure emerges. As shown in Figure 2, the *E-R* structure rapidly changes into a scale-free, with

a scaling parameter of about 2. As temperature decreases, the structure begins to converge to the *WTA* structure characterized by a high value of the scaling parameter. In particular, a homogeneous structure emerges, with the presence of hubs (i.e., nodes with high degree). At the end of the cooling process, few nodes have a very high degree ($\sim n$) and the remaining nodes have low degree. Other analyses about the connection between classical random and scale-free networks have been reported in [8]. In the cited paper, the authors show that, for cold regime, their network is scale-free, but as the temperature increases the network loses its metric structure and its hierarchical heterogeneous organization, becoming a classical random network. Considering that many real complex networks are scale-free while others are not (see for example [16]), we deem that the proposed fermionic model can be considered a good candidate for representing their evolution, at low and high temperatures. As shown in Figure 3, we analyzed also the dynamics of particles during the cooling process. We observed that the number of particles changing position is very high from the first time step, but it decreases rapidly.

6 Conclusions

In this work, we define a fermionic network model that allows to represent complex networks as quantum gases. Using this model, we show that network evolution is a temperature-dependent process characterized by three main phases: classical random, scale-free and *WTA*. The transition from classical random to scale-free networks mimics a cooling process of a physical system and, when a winner-takes-all structure is obtained, the system achieves equilibrium, despite the non-equilibrium nature of the network evolution. We deem that mapping complex networks to quantum systems, can be useful to analyze some of their properties. Furthermore, this model allows also to represent competitive dynamics among nodes. In particular, nodes are characterized by a fitness parameter and, during cooling processes, only few of them gain the majority of links. As for future work, we are planning to develop a model in which the fitness of each node can vary over time.

References

1. Albert, R., Barabasi, A.L.: Statistical Mechanics of Complex Networks. Rev. Mod. Phys. 74, 47–97 (2002)
2. Guimer, R., Danon, L., Diaz-Guilera, A., Giralt, F., Arenas, A.: Self-similar community structure in a network of human interactions. Phys. Rev. E Stat. Nonlin. Soft. Matter Phys. 68 (2003)
3. Sporns, O., Chialvo, D.R., Kaiser, M., Hilgetag, C.C.: Organization, development and function of complex brain networks. Trend in Cognitive Sciences 8(9) (2004)
4. Erdos, P., Renyi, A.: On the Evolution of Random Graphs. pubblication of the mathematical institute of the hungarian academy of sciences, pp. 17–61 (1960)
5. Watts, D.J., Strogatz, S.H.: Collective dynamics of "small-world" networks. Nature, 40–442 (1998)

6. Bianconi, G., Barabasi, A.L.: Bose-Einstein Condensation in Complex Networks. Physical Review Letters 86, 5632–5635 (2001)
7. Bianconi, G.: Quantum statistics in complex networks. Physical Review E 66 (2002)
8. Krioukov, D., Papadopoulos, F., Kitsak, M., Vahdat, A., Boguna, M.: Hyperbolic Geometry of Complex Networks. Physical Review E (2010)
9. Shen, Y., Zhu, D.-L., Liu, W.-M.: Fermi–Dirac Statistics of Complex Networks. Chinese Physics Letters 22, 5 (2005)
10. Baronchelli, A., Catanzaro, M., Pastor-Satorras, R.: Bosonic reaction-diffusion processes on scale-free networks. Physical Reviw E 78 (2008)
11. Perseguers, S., Lewenstein, M., Acin, A., Cirac, J.I.: Quantum complex networks. Nature Physics 6 (2010)
12. Huang, K.: Statistical Mechanics. John Wiley and Sons (1987)
13. Hartonen, T., Annila, A.: Natural Networks as Thermodynamic Systems. Complexity (2012)
14. Newman, M.E.J., Watts, D.J., Strogatz, S.H.: Random Graphs Models of Social Networks. PNAS 99, 2566–2572 (2002)
15. Clauset, A., Shalizi, C.R., Newman, M.E.J.: Power-Law distributions in empirical data. SIAM Review 51, 661–703 (2009)
16. Newman, M.E.J.: The structure and function of complex networks. SIAM Reviews 45, 167–256 (2003)

New Selection Schemes in a Memetic Algorithm for the Vehicle Routing Problem with Time Windows

Jakub Nalepa[1] and Zbigniew J. Czech[1,2]

[1] Silesian University of Technology, Gliwice, Poland
{jakub.nalepa,zbigniew.czech}@polsl.pl
[2] University of Silesia, Sosnowiec, Poland

Abstract. This paper presents an extensive study on the pre- and post-selection schemes in a memetic algorithm (MA) for solving the vehicle routing problem with time windows. In the MA, which is a hybridization of the genetic and local optimization algorithms, the population of feasible solutions evolves with time. The fitness of the individuals is measured based on the fleet size and the total distance traveled by the vehicles servicing a set of geographically scattered customers. Choosing the proper selection schemes is crucial to avoid the premature convergence of the search, and to keep the balance between the exploration and exploitation during the search. We propose new selection schemes to handle these issues. We present how the various selection schemes affect the population diversity, convergence of the search and solutions quality. The quality of the solutions is measured as their proximity to the best currently-known feasible solutions. We present the experimental results for the well-known Gehring and Homberger's benchmark tests.

1 Introduction

The vehicle routing problem with time windows (VRPTW) is a bi-criterion discrete optimization problem belonging to the NP-hard class. It consists in finding a set of feasible routes for a set of homogenous vehicles servicing the customers dispersed on the map. The customers must be visited within their time windows and the vehicle capacities cannot be exceeded. The VRPTW is a hierarchical optimization problem. The first objective is to minimize the number of routes whereas the second one is to minimize the total traveled distance. The VRPTW has numerous practical applications including the bus planning, parcels and food delivering, cash delivering to banks and ATM terminals and many more.

A number of exact and approximate algorithms for solving the VRPTW have been proposed in the recent years due to its wide practical applicability. An extensive review of the exact methods can be found in Kallehauge [1]. The current state-of-the-art heuristic algorithms to solve the VRPTW can be divided into the construction and improvement methods. In the construction heuristic algorithms the customers are inserted iteratively into a partial solution, until

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 396–405, 2013.

a feasible set of routes is generated. Several construction heuristics have been proposed [2]. In the improvement heuristics an initial feasible solution is modified to decrease the number of vehicles and the traveled distance. An example of such approach is given in Potvin and Rousseau [3]. In the metaheuristics, the mechanisms to explore and to exploit the search space are applied. The feasible solutions may be perturbed and deteriorated to escape the local minima. A survey on the metaheuristic algorithms can be found in Bräysy and Gendreau [4].

The memetic algorithms (MAs) are the population-based approaches combining the evolutionary algorithms for the exploration of the search space S with the local optimizations for the exploitation of S. The EAX-based MA has been proposed by Nagata and Bräysy [5] and later improved [6, 7]. In the work reported here we use the parallel heuristic algorithm [7] to generate a population of solutions. We study the influence of various pre- and post-selection schemes on the performance of a sequential MA [7] to minimize the travel distance, apply new schemes to diversify the search and evaluate them experimentally.

Choosing the proper selection schemes and measuring the population diversity for the population-based approaches appear crucial to avoid the premature convergence of the search. The most common measures of the diversity include pair-wise Hamming distance calculations in the genotypic space, calculating the fitness standard deviation in the phenotypic space or measuring the number of distinct individuals in a population. The moment of inertia method for estimating the diversity has been introduced by Morrison and De Jong [8].

The paper is organized as follows. The VRPTW is formulated in Section 2. The memetic algorithm and the selection schemes are presented in Section 3. Section 4 discusses the experimental results. Conclusions and directions for the future work are given in Section 5.

2 Problem Formulation

The VRPTW is defined on a directed graph $G = (V, E)$ with a set V of $N + 1$ vertices representing the customers and the depot, along with a set of edges $E = \{(v_i, v_{i+1}) | v_i, v_{i+1} \in V, v_i \neq v_{i+1}\}$ representing the connections between the travel points. Each vehicle starts and finishes at the depot v_0. The travel costs are given as $c_{i,j}$, where $i \neq j$, $i, j \in \{0, 1, ..., N\}$. The non-negative customer demands d_i, $i \in \{0, 1, ..., N\}$, where $d_0 = 0$ and the time windows $[e_i, l_i]$, $i \in \{0, 1, ..., N\}$ are defined. The customers have their service times s_i, $i \in \{1, 2, ..., N\}$. A fleet of vehicles with a constant capacity Q is given. Let K denote a size of the fleet. The route is defined as a set of customers serviced by a single vehicle $(v_0, v_1, ..., v_{n+1})$, where $v_0 = v_{n+1}$. A solution σ (a set of routes) is feasible if (i) the total amount of goods delivered to the customers within each route does not exceed the vehicle capacity Q, (ii) the service of a customer v_i is started before the time window $[e_i, l_i]$ elapses, (iii) every customer v_i is serviced in exactly one route, and (iv) every vehicle leaves and returns to the depot within its time window $[e_0, l_0]$. The penalty function $F(\sigma)$ is introduced to determine the degree of the constraint violations in the infeasible solutions [5].

The primary objective of the VRPTW is to minimize the fleet size K. The lower bound for the number of vehicles is given by $K_{min} = \lceil D/Q \rceil$, where $D = \sum_{i=1}^{N} d_i$. The secondary objective is to minimize the total distance $T = \sum_{i=1}^{K} T_i$, where K is the number of routes and T_i is the distance of the *i-th* route.

3 Memetic Algorithm and Selection Schemes

The initial population of size N of feasible solutions (individuals) is generated using a construction parallel heuristic algorithm based on the heuristics proposed by Nagata and Bräysy [5] (Fig. 1, lines 1–2). In this section we present the outline of the MA for the total distance minimization (its detailed description is given in [7]). We propose some new selection schemes used in the MA. The pre-selection scheme defines how the mating pool is generated and how the pairs of individuals are chosen for the edge-assembly crossover (EAX) operator. In the post-selection step we determine which individuals from the *i-th* generation and the children pool survive and form the $(i+1)$-*th* generation.

3.1 Memetic Algorithm Outline

The minimal number of routes K in a feasible solution is determined at first using the parallel heuristic algorithm [7] (Fig. 1, line 1). Then, N individuals with the number of routes K are generated (line 2). The fitness $\eta(p_i)$ of an individual p_i corresponds to the travel distance T_i (K is constant within a population), thus a lower distance T_i is considered as a better fitness $\eta(p_i)$.

The individuals from the *i-th* generation are used to create the $(i+1)$-*th* generation. Once the chromosomes p_A and p_B are selected, they are recombined using the EAX operator (line 7) to generate N_c offspring solutions, each composed of K routes (Fig. 2). If the child solution p_c is infeasible, i.e. if $F(p_c) > 0$, where $F(p_c)$ is the penalty function of p_c, then the feasibility of the solution is restored by performing the local repairing moves (Fig. 1, line 8). The moves, either inter- or intra-route, are based on exchanging the edges between two routes r_i and r_j, where $i \neq j$, or within a single route r_i. Only moves decreasing the value of $F(p_c)$ are performed. Finally, if the solution p_c is feasible then it is improved by at most I perturbation moves (only the feasible moves decreasing the total distance are accepted) (line 9). The best child p_c^b is chosen for each pair of parents (lines 10–12). The offspring solutions inherit the features of both parents p_A and p_B, thus the post-selection scheme has a strong impact on the population diversity and the search convergence. Depending on the approach, the $(i+1)$-*th* generation is a set of N best children or a mixed set of individuals belonging to the *i-th* generation and the children pool (line 15). The algorithm continues until the termination condition is met (line 16). Finally, the best individual in the last population is returned (line 18).

3.2 Selection Schemes

If the population is saturated with similar individuals then the probability of exploring the search space S drops. Thus, escaping the local minima becomes

```
1: Determine a number of routes K using a parallel heuristic algorithm;
2: Generate an initial population of solutions (individuals) of size N (each solution
   consists of K routes);
3: while not finished do
4:    Determine N reproduction pairs (p_A, p_B); {Pre-selection}
5:    for each pair (p_A, p_B) do
6:       for i ← 1 to N_c do
7:          p_c ← EAX(p_A, p_B);
8:          p_c ← Repair(p_c);
9:          p'_c ← LocalSearch(p_c);
10:         if η(p'_c) > η(p^b_c) then
11:            p^b_c ← p'_c;
12:         end if
13:      end for
14:   end for
15:   Form the next population; {Post-selection}
16:   finished ← VerifyTerminationCondition();
17: end while
18: return best individual in the last population;
```

Fig. 1. The MA for the total distance minimization

impossible and the population should be re-generated or perturbed if it is in the diversity crisis. This induces a trade-off between the exploration and the exploitation of S. Here we present the selection schemes explored in this work:

1. **AB-selection**. Each individual p_i, $i = 1, 2, \ldots, N$, is selected as p_A at first. Then, the individual p'_i is chosen as p_B, where $p_A \neq p_B$. Each individual p_i can be selected once as p_A and once as p_B. The parent p_A is replaced by the best offspring solution p_c only if $T_c < T_A$.
2. **High-low fit**. The original version of this method was proposed by Elamin in [9]. The parent p_A is selected from the $c_h \cdot N$ fittest individuals, where c_h, $0 < c_h < 1$, is the high-low selection coefficient. The second parent is drawn from the less-fitted part $(1 - c_h) \cdot N$ of the sorted population. The N fittest individuals survive. We propose to sort a set of individuals of size $2N$ containing N offspring solutions and the individuals from the *i-th* generation, and select the N_d, $N_d \leq N$, fittest, but distinct individuals to form the $(i+1)$-*th* generation. If $N_d < N$ then remaining $N - N_d$ individuals, starting from the fittest, are selected and perturbed by at most I_p perturbation moves (if the feasible ones exist) to increase the population diversity.
3. **Sexual selection**. In this scheme introduced by Goh et al. [10] the sex of each individual is determined randomly at first – we create $N/2$ females and $N/2$ males. Then, each female is selected twice as p_A for N EAX pairs. The tournament selection with the tournament size 2 is applied to select a male as p_B for each female. The N best children form a new population.
4. **Enhanced truncation**. In the standard truncation scheme [11], the parents p_A and p_B are drawn from the $c_t \cdot N$ fittest individuals, where c_t, $0 < c_t < 1$,

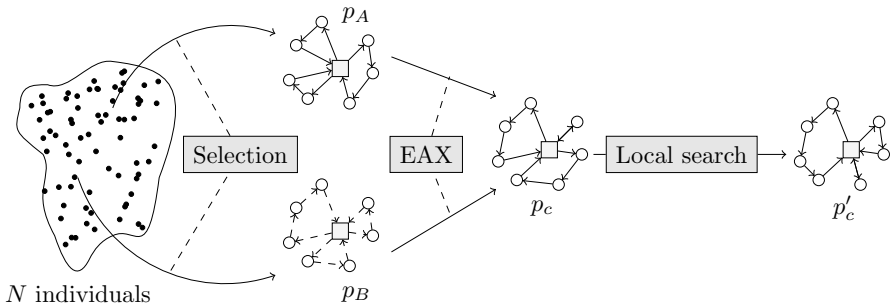


Fig. 2. Generation of a child solution

is the truncation coefficient, and the child population is composed of the best child solutions. The enhanced truncation was proposed by Kawulok and Nalepa [12]. The $c_r \cdot N$ pairs p_A and p_B are selected from the $c_e \cdot N$ fittest individuals, where c_r, $0 < c_r < 1$, is the reproduction coefficient and c_e, $0 < c_e < 1$, is the enhanced truncation coefficient. The best offspring solutions for $c_r \cdot N$ pairs of parents survive to the next generation. The remaining $N - c_r \cdot N$ individuals are generated randomly, what simulates additional mutation within the population of size N.

5. **Adaptive selection**. In this selection scheme we apply the enhanced truncation scheme for the exploration of the search space with the elitist approach. In the elitist approach [13], the $c_b \cdot N$, $0 < c_b < 1$, best individuals are copied from the *i-th* generation to the $(i + 1)$-*th* generation, where c_b is the elitist coefficient. If the best individual in the population is not improved for the g_s consecutive generations, then the selection scheme is changed to the AB-selection to exploit the search space.

4 Experimental Results

The proposed selection schemes have been used in the MA to solve the Gehring and Homberger's (GH) benchmark tests containing 200 customers. The GH tests are divided into six groups, namely: C1, C2, R1, R2, RC1 and RC2. Clusters of customers are given in the C class problems. Customers are randomly scattered around the map in case of the R class. The RC class contains both clustered and randomized customers. Each class is further divided into two subclasses containing the problems with small vehicle capacities and short time windows (C1, R1, RC1) and these with larger vehicle capacities and wider time windows (C2, R2, RC2). Each subclass contains 10 problem instances.

The MA was implemented in C++ and the experiments were performed on a computer equipped with an Intel Core i7 2.3 GHz (16 GB RAM) processor. The population was composed of $N = 80$ individuals for each test. The MA parameters were tuned experimentally to the following values: $N_c = 20$, $c_h = 0.5$,

Table 1. Average number of distinct individuals n vs. generation g: (a) AB-selection, (b) High-low fit, (c) Sexual selection, (d) Enhanced truncation, (e) Adaptive selection, for GH subclasses C1, C2, R1, R2, RC1, RC2 (200 customers)

	$g=1$					$g=50$					$g=100$					$g=150$					$g=200$				
	(a)	(b)	(c)	(d)	(e)	(a)	(b)	(c)	(d)	(e)	(a)	(b)	(c)	(d)	(e)	(a)	(b)	(c)	(d)	(e)	(a)	(b)	(c)	(d)	(e)
C1	74	74	72	73	74	58	77	52	43	48	44	78	46	49	47	44	79	46	52	44	41	78	43	52	45
C2	79	80	80	78	79	47	80	46	41	45	42	80	38	44	44	41	80	39	39	41	43	80	41	41	43
R1	80	80	80	79	80	64	80	56	46	45	42	80	45	53	44	39	80	38	54	39	43	80	38	54	41
R2	80	80	80	80	80	51	80	48	39	46	35	80	34	46	34	35	80	34	37	31	35	80	36	40	31
RC1	80	80	79	80	80	65	80	56	47	39	50	80	42	50	42	43	80	40	51	41	45	80	40	60	38
RC2	80	80	78	79	80	55	80	55	33	55	34	80	33	39	34	34	80	35	42	31	33	80	35	42	34

$I = I_p = 300$, $c_r = 0.9$, $c_e = 0.5$, $c_b = 0.05$, $g_s = 10$, $g_{max} = 200$, where g_{max} is the maximal number of generations. Each GH test was run 5 times using each selection scheme to solve each problem instance. The averages of the distances T of the best individuals in the populations, of the numbers of distinct ones n, $n \leq N$, and of the standard deviations s of the total distances were computed. Then, the results were averaged for each subclass (C1, C2, R1, R2, RC1, RC2). The average numbers of distinct individuals are given in Tab. 1, the average travel distance of the best individual and the standard deviation of the travel distance are shown in Fig. 3(a–f) and Fig. 4(a–f) respectively. The best-known travel distances are averaged for each subclass and presented in Fig. 3. The number of routes K was equal to the best-known number K_b for each GH test[1].

The experiments showed that the GH tests with wide time windows and large vehicle capacities (C2, R2, RC2) can be solved in relatively small number of generations (Fig. 3(b,d,f)). It is easy to note that the average travel distances drive towards the average best-known distance quickly (in approx. 80 generations). The GH tests belonging to the C1, R1 and RC1 subclasses with short time windows and smaller vehicle capacities turned out to be harder to solve to good accuracy in short time (Fig. 3(a,c,e)). The main objective was to investigate how the schemes discussed in Section 3.2 influenced the search convergence and the population diversity. We measured the population diversity by n and s.

At the beginning of each search process, the population was composed of a large number ($n \approx N$) of distinct solutions (Tab. 1, $g = 1$). The n was subsequently decreased during the algorithm execution for most of the selection schemes. In case of the high-low selection (Tab. 1(b)) the additional perturbing moves were applied and n remained at constant level. It is worth noting that considering only the number of distinct individuals for the estimation of the population diversity may be misleading. It can been seen in Fig. 4(a–f) and Tab. 1 that the standard deviation for the AB-selection scheme for $100 < g < 200$ is close to 0. Moreover, for this range we have $33 \leq n \leq 50$, thus $0.41N \leq n \leq 0.63N$. This indicates that the population was saturated with similar (well-fitted) individuals as a result of intensive exploitation of the search space. The large values of s were observed for the sexual and the enhanced truncation schemes.

[1] The best-known solutions of the GH benchmark tests are published on the Sintef website: `http://www.sintef.no/Projectweb/TOP/VRPTW/Homberger-benchmark/`

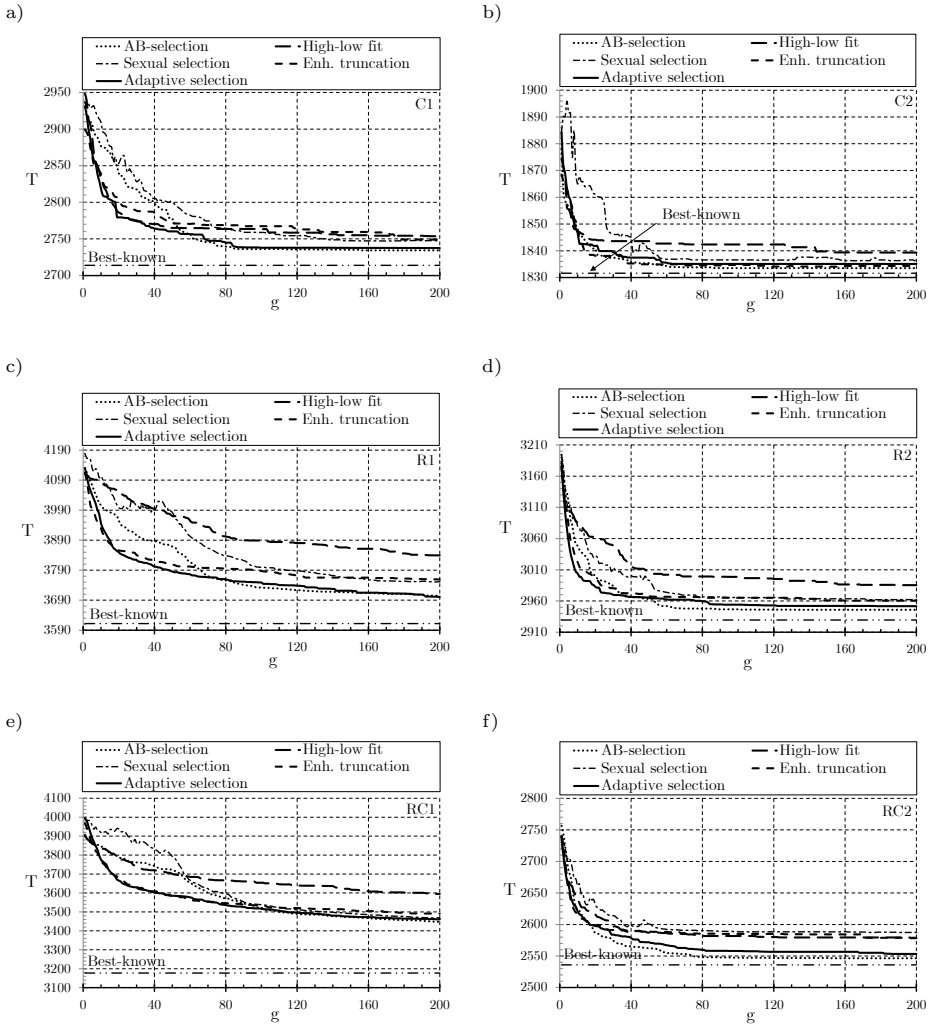


Fig. 3. Average travel distance T of the best individual vs. generation g for GH subclasses C1, C2, R1, R2, RC1, RC2 (200 customers)

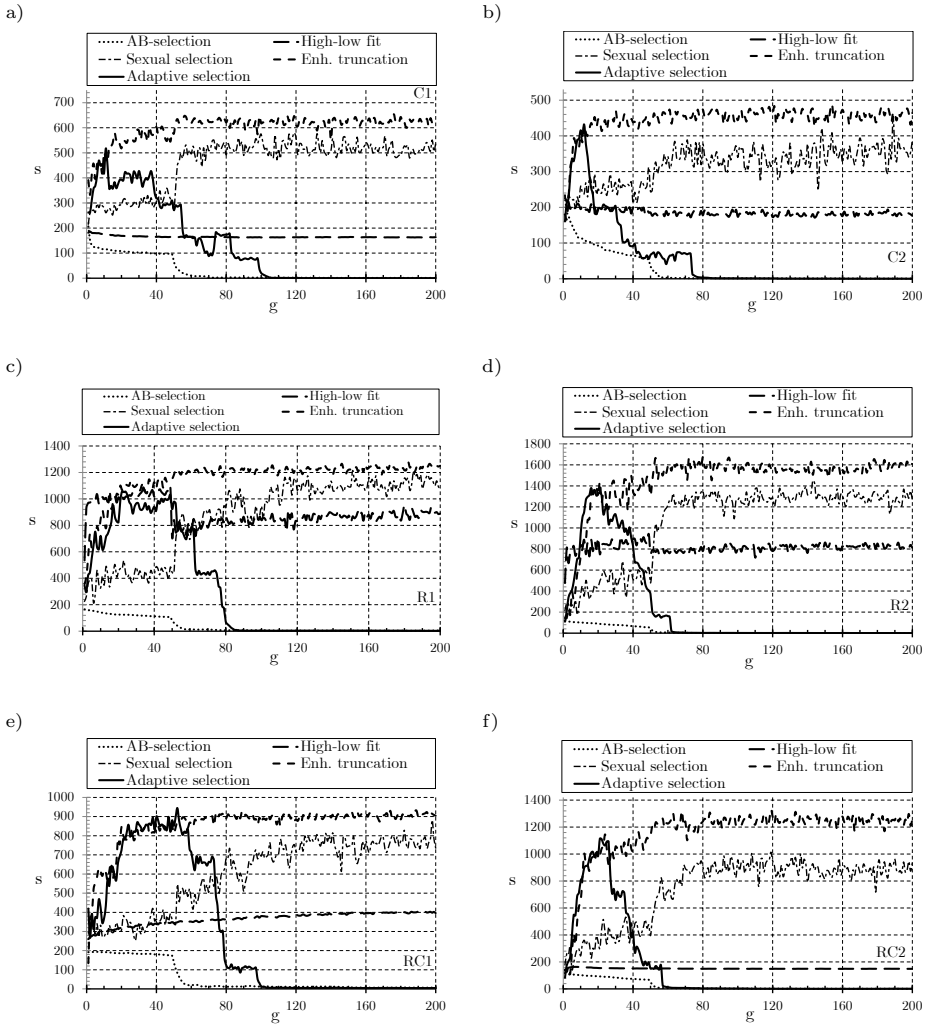


Fig. 4. Average standard deviation s of the travel distance T vs. generation g for GH subclasses C1, C2, R1, R2, RC1, RC2 (200 customers)

In case of the adaptive selection, the standard deviation was dependent on the search stage. During the initial exploration of the search space, the value of s was quite large. Once the scheme switched from the elitist enhanced truncation to the AB-selection, the s started decreasing (the exploitation stage). Finally, the standard deviation approached zero and the population was composed of well-fitted individuals. The average number of distinct individuals n became similar

(Tab. 1) for the AB-selection and the adaptive selection during the exploitation ($g \gtrsim 100$).

As mentioned earlier, the GH tests can be divided into two groups: tests that can be solved fast to good accuracy (C2, R2, RC2) and those converging slower (R1, R2, RC1). The intensive exploitation yielded approaching the best solutions for the tests of the first group (Fig. 3(b,d,f)). The experiments indicated that the adaptive and AB-selection schemes proved to be the best among the schemes under investigation, in terms of convergence and accuracy of final results. These selection schemes guided the search towards the best solutions and s started decreasing relatively fast, thus the population contained well-fitted and similar individuals. The application of the other selection schemes resulted in increasing the population diversity, however the search converged slower. In case of the more difficult tests (C1, R1, RC1), the speed of converging to the best-known results dropped for the AB-selection. Applying the selection schemes that ensured the large values of s (the high-low fit and the enhanced truncation) led to obtaining the better solutions faster (Fig. 3(a,c,e)). However, they were not further improved significantly. Although the AB-selection scheme converged slower than the high-low fit and the enhanced truncation, the steady-state solutions were of higher accuracy. It is worth noting that the adaptive selection scheme outperformed other schemes in terms of the convergence speed (without worsening the final solutions accuracy). It combined the advantages of the initial search space exploration and its further exploitation. It can be seen that in case of the sexual selection it may happen that the best individuals in the i-*th* generation do not survive to the $(i+1)$-*th* generation, thus the best-individual fitness in the $(i+1)$-*th* generation can decrease with respect to the i-*th* generation.

5 Conclusions and Future Work

We proposed new selection schemes for the memetic algorithm to solve the vehicle routing problem with time windows. We showed how the choice of a scheme affects the final solutions accuracy, the convergence speed and the diversity of the population. During the experiments the Gehring and Homberger's benchmark tests containing 200 customers were used. The adaptive and the AB-selection schemes proved to be the best among the considered ones. The adaptive selection scheme outperformed other schemes in terms of the search convergence speed. The travel distances of the best individuals were approximately the same for the AB-selection and the adaptive selection schemes, however the former required much larger number of generations to converge to solutions of high quality.

Our ongoing research includes performing the time complexity analysis and running the full GH tests using the parallel version of the MA to compare the execution times in practice. The influence of the algorithm settings on its performance is to be clarified. Also, our aim is to conduct the sensitivity analysis for the proper adjustment of the settings allowing for their automatic tuning. Finally, we want to expand the schemes to explicitly utilize the diversity measures [14].

Acknowledgments. We thank the following computing centers where the computations of our project were carried out: Academic Computer Centre in Gdańsk TASK, Academic Computer Centre CYFRONET AGH, Kraków, Interdisciplinary Centre for Mathematical and Computational Modeling, Warsaw University, Wrocław Centre for Networking and Supercomputing.

References

1. Kallehauge, B.: Formulations and exact algorithms for the vehicle routing problem with time windows. Comput. Oper. Res. 35(7), 2307–2330 (2008)
2. Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. Trans. Sc. 39(1), 104–118 (2005)
3. Potvin, J.Y., Rousseau, J.M.: An exchange heuristic for routing problems with time windows. J. of the Oper. Res. Society 46, 1433–1446 (1995)
4. Bräysy, O., Gendreau, M.: Vehicle Routing Problem with Time Windows, part II: Metaheuristics. Trans. Sc. 39(1), 119–139 (2005)
5. Nagata, Y., Bräysy, O., Dullaert, W.: A penalty-based edge assembly memetic algorithm for the vehicle routing problem with time windows. Comput. Oper. Res. 37(4), 724–737 (2010)
6. Blocho, M., Czech, Z.J.: A Parallel EAX-based Algorithm for Minimizing the Number of Routes in the Vehicle Routing Problem with Time Windows. In: HPCC-ICESS, pp. 1239–1246 (2012), `http://sun.aei.polsl.pl/~zjc/pub.html`
7. Nalepa, J., Czech, Z.J.: A parallel heuristic algorithm to solve the vehicle routing problem with time windows. Studia Informatica 33(1), 91–106 (2012), `http://sun.aei.polsl.pl/~zjc/pub.html`
8. Morrison, R.W., De Jong, K.A.: Measurement of Population Diversity. In: Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E., Schoenauer, M. (eds.) EA 2001. LNCS, vol. 2310, pp. 31–41. Springer, Heidelberg (2002)
9. Elamin, E.E.A.: A proposed genetic algorithm selection method. In: NITS (2006)
10. Goh, K., Lim, A., Rodrigues, B.: Sexual selection for genetic algorithms. Artif. Intell. Review 19, 123–152 (2003)
11. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm. Evol. Comp. 1, 25–49 (1993)
12. Kawulok, M., Nalepa, J.: Support Vector Machines Training Data Selection Using a Genetic Algorithm. In: Gimel'farb, G., Hancock, E., Imiya, A., Kuijper, A., Kudo, M., Omachi, S., Windeatt, T., Yamada, K. (eds.) SSPR&SPR 2012. LNCS, vol. 7626, pp. 557–565. Springer, Heidelberg (2012)
13. Murata, T., Ishibuchi, H., Tanaka, H.: Multi-objective genetic algorithm and its applications to flowshop scheduling. Comput. Ind. Eng. 30(4), 957–968 (1996)
14. Lü, Z., Glover, F., Hao, J.K.: A hybrid metaheuristic approach to solving the UBQP problem. E. J. of Op. Res. 207(3), 1254–1262 (2010)

Classification Based on the Self-Organization of Child Patients with Developmental Dysphasia

Jana Tuckova[1], Josef Vavrina[1], Jan Sanda[2], and Martin Kyncl[2]

[1] Czech Technical University, Faculty of Electrical Engineering
tuckova@fel.cvut.cz, pepa.vavrina@gmail.cz
[2] Charles University in Prague, 2nd Faculty of Medicine, Department of Radiology
{jan.sanda,martin.kyncl}@fnmotol.cz

Abstract. Involvement of mathematical and engineering methods in medicine makes it possible to perform research into processes in the human body by non-invasive methods. Our team cooperates with neurologists in the domain of developmental dysphasia. We search for correlations between the results of EEG, magnetic resonance (MR) tractography, speech signal analysis, clinical speech therapy and psychology. Our aim is to verify a hypothesis of the possibility of classifying and visual representing changes in pathological speech by means of artificial neural networks. This contribution concentrates on one part of this research: disordered children's speech analysis and results from MR tractography. We try to divide the patients into three groups according to disorder relevance. For classification, we use PCA and SSOM. Evaluation of the results and preparation of a software pack with a user-friendly interface can facilitate the emergence of disease monitoring and improve the quality of therapy.

Keywords: SSOM, PCA, MR tractography, developmental dysphasia, pathological children speech.

1 Introduction

At present, mathematical-engineering methods are used in many medical disciplines. Improvement of diagnostic and therapeutic methods is the result of rapid advances in informatics, new technologies, and device techniques. We wanted to utilize our years of experience about speech from the beginning of our collaboration with neurologists, because language impairment can be caused by a number of brain disorders. For eight years, collaboration has been underway between Laboratory of the Neural Network Applications (CTU FEE in Prague) and the Paediatric neurology department from the 2nd Faculty of Medicine of CU in Prague. Recently, collaboration has included the Department of Radiology from the same clinic as well.

We search for relations between speech signal characteristics and cerebral activity, which are important for an understanding of pathological speech demonstration in joint research. Pathological speech is one of the symptoms of the neurological disorder named Developmental Dysphasia (DD). An implication of early cerebral

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 406–416, 2013.

damage, DD is a central disorder of speech processing which affects all speech modalities (phonetic-phonologic, morphological-syntactic, lexical-semantic and memory) and also other developmental aspects of the child's personality. The disorder interferes with speech zones (Broca's area and Wernicke's center) of the developing child's brain in the prenatal period, and patients show a deficiency of speech production and understanding. DD is found to affect about 5% of the paediatric population [1]. This contribution describes the results of the analysis of disordered speech from 14 children with DD in the pilot study.

To the best of our knowledge, our investigation is the first project in a search for correlations between computer child speech processing and MR tractography. Nerologists, speech therapists and other interested experts study the problem of pathological speech as a consequence of cerebral activity from a language-based point of view [2], [3], not from the engineering approach of speech signal processing. Here, MR tractography data are processed and analysed by standard statistical methods, usually see [4] or by Support Vector Machines (SVM) [5]. The first method describes a problem similar to our research, while the second one is often used for patients with Alhei-mer's disease. Our approach is different in that we proceed from our experiences with computer signal processing and from KSOM (Kohonen Self-Organizing Maps) or SSOM (Supervised Self-Organizing Maps) applications [6] for data classification of child neurological patients [1].

2 Hypothesis

Our point of departure is the hypothesis that the underlying factor is the changes of movement of speech organs in the articulation of children with neurological disorders, which in turn impacts formant generation. Shifting of formants in the frequency spectrum is one effect of this process. Formant location is primarily characteristic for vowel generation, but also has an influence on the co-articulation surface at dislocation and pronunciation of consonants. We have supposed that the movement of formants can be revealed through SSOM [1]. One advantage of these maps is their very strong visualisation ability. Graphic representation of phonemes (cluster visualization) by graphic display in 2D is suitable for diagnosis of disease by medical practitioners. Analysis of the layout and movement of the features in SSOM can uncover one of the symptoms for neurological disease identification. The other advantage common to all ANN paradigms is the robustness of the solution for real methods. The method using principle component analysis (PCA) [7] was created based on promising results of SSOM. The idea of PCA was based on the same principle, and tries to use the standard methods for comparison. It is necessary to find out the optimal principal components, whether using Mel-frequency-cepstral coefficients (MFCC) or LPC and many other parameters of the model, which detects degree of DD. In addition, certain parameters of MR tractography (Diffusion Tensor Imaging - DTI method) [8] can be used for patient classification. The first idea is to separate all obtained data into 3 groups by KSOM, using all parameters. The second idea is to choose only those data that are assumed to correspond with the DD (according to doctors) and apply the same method (KSOM). The third method is to classify data into 3 groups entirely from the original parameters – choosing the promising ones.

3 Methods

KSOM and SSOM were used for the first experiments [1], [6]. Both types of SOMs are based on clustering. These maps and their subsequent visualisation serve to monitor the progress of trends and the magnitude of the degree of impairment. The object of the search is for correlations between results from computer speech analysis of the children with DD, and the other methods surveyed in the project (MR tractography in this contribution). ANNs were selected for the reason of their notable robustness and very good ability to perform data visualisation, hence they can also process less qualitative signals. The quality of recorded utterances is impacted by the real environment during the recording, namely non-professional speakers, environments with strong noise, even children's age). Data visualization enables the following of changes at the generation of formant frequencies, which has an influence on input vector mapping into particular clusters. SSOMs are utilized to find identical characteristic features in utterances. Features in the signal that are spatially or temporally adjacent are represented by patterns. If the maps are trained with healthy children's utterances, the patterns represent the distribution of the feature in their speech. The differences between healthy and DD children could be enumerated in proportion to the progress of treatment being described: in cases of effective therapy, the differences tend to decrease. SSOMs create clusters of all input vectors with common or close labels. Clusters characterize particular phonemes and when allocated to the map they show partial numbers of the dominant characteristics (frequency distribution of particular phone-mes) over one training process. We can divide patients into three groups according to the classification success. These groups correspond with relevance to the disorder to the described task (light, middle and severe DD). A characteristic of the task consists in the accentuation of errors and deviations of standard values among healthy children.

3.1 Analysis of The Children's Speech Signal

We study the ability of phoneme classification, regarding phonemes that are pronounced separately and as well as part of words. Recorded speech is pre-processed by methods frequently used in a speech signal processing in engineering applications [9]. SSOM [6], [10] and PCA [7], [11] are used for phoneme classification. Our first experiments [1] provided proof for the veracity of our previous hypothesis. The SSOM consists of m units located on a regular, low-dimensional grid. Each unit is connected to a number of neighbouring units with a neighbourhood relation. Supervised learning means that the input vector is formed of two parts, $\mathbf{x_0}$ and $\mathbf{x_c}$, where $\mathbf{x_0} = [x_{01}, x_{02},\ldots, x_{0n}]^T$, $\mathbf{x_0} \in R^n$ is an original input vector of dimension n and $\mathbf{x_c} = [x_{c1}; x_{c2},\ldots, x_{ck}]^T$, $\mathbf{x_c} \in R^k$, k is assigned as a known class of x_c (supervisor) in a training set (indication of vowels in our experiments). Each element of vector x_c represents one of k classes. A new vector $\mathbf{x} = [\mathbf{x_0}, \mathbf{x_c}]^T$, $x \in R^{n+k}$ will have a dimension $(n + k)$, which is valid for a prototype vector $\mathbf{m} = [m_1,m_2,\ldots,m_{n+k}]^T$, $m \in R^{n+k}$ as well. During the classification of an unknown input vector x, only its x_0 part was compared with the corresponding part of the prototype vectors. The class of each unit is found by taking the maximum over these added elements. Spoken speech is a time-dependent sequence of phonemes, so as a result it is necessary to process the input data to ANN in

a batch form: this method is significantly faster and does not require any specification of a learning-rate factor. New prototype vectors are calculated as a weighted average of the input vectors, where the weight of each input vector is the neighbourhood function value $h_{i,m*(j)}$ at its winner $\mathbf{m}*\mathbf{j}$:

$$m^{i}(t+1) = \frac{\sum_{j=1}^{N} h_{i,m*j}(t) x_j}{\sum_{j=1}^{N} h_{i,m*j}(t)}, \quad (1)$$

where t is the number of iterations, x_j is the input vector, N is the number of input vectors. The most usual neighbourhood function is the Gaussian one. During the training, the map adjusts to the data by adapting the prototype vectors. After the training phase, each map unit will be automatically assigned the most probable labels which contain the information about classes of the training set. The labelling process is based on the searching maximum of the second part of the training data set. Finally, the second part of the training data set is removed, as the elements were useful only for labelling the map units. The method has been used for training of maps for each patient.

The PCA model is created in steps:

1. Calculate MFCC and LPC coefficients from utterances of all healthy children (after vowel extraction).
2. Use PCA to create a new dimension space based on principal components (PCs).
3. Split data into two groups – the validation data set and the training data set.
4. Now the data model is created, and it is possible to plot it into 2D (according to the selection of principal components).
5. Find out the parameters of the model (cluster information for each vowel).

The parameters of the model contain the mean value of the cluster (centroid) and its covariance matrix, which is represented by Gaussian distribution. Table 1 is an example of a child, in which the main diagonal represents the correct classification (values in rows represent actual data and values in columns represent output of VD). Each segment is classified as a vowel based on the closest centroid (the highest probability among Gaussian distributions). Table 2 represents the gravity of classifications based on the vocalic triangle [9] – value 1 for correct classification, value 5 for misclassification among close vowels and value 25 for misclassifications among distant vowels. The gravity values in Table 2 are defined from our linguistic experience.

Table 1. Classification [%]

	a	e	i	o	u
a	99.8	0.2	0	0	0
e	4,6	95,4	0	0	0
i	0	0	99.8	0	0,2
o	0	0	0,7	65.1	**34.2**
u	0	0	**9.1**	0	90,9

Table 2. Gravity of misclassification

	a	e	i	o	u
a	1	5	25	5	25
e	5	1	5	25	25
i	25	5	1	25	25
o	5	25	25	1	**5**
u	25	25	**25**	5	1

Table 3. Classification with gravity

	a	e	i	o	u
a	1	0	0	0	0
e	0.2	1	0	0	0
i	0	0	1	0	0
o	0	0	0,2	0,7	**1.8**
u	0	0	**2.3**	0	0.9

Table 3 is the multiple of values in tables 1 and 2. The misclassification between neighbour vowels in the vocalic triangle is not as serious (these misclassifications are usual as well among healthy children). By contrast, the misclassification between \i\ and \u\ (see 4.1.) is one of the DD symptoms (note the values in bold in the tables), see [1]. Finally, the summarization of all numbers in Table 3 is the "sickness" parameter. The "sickness" parameter is determined for each child in the validation set (healthy children). The parameters of the model were determined according to the "sickness" parameter, in which the best model has the lowest "sickness parameter". The best vowel detector (VD) model was created in the following steps:

1. set/reset all model parameters,
2. create model,
3. determine the mean of the sickness parameter among all children in validation set,
4. remember the value of the sickness parameter and go to step 1,
5. if all parameters are tested, choose the model with the lowest sickness parameter.

Formation of the Utterance Database

The database contains the records of utterances of 44 female and 28 male healthy children and 67 children with DD, and was built for an assessment of a degree of a speech impairment. The children's age is from 4 to 10 years [12]. Recordings of the utterances were realized at kindergartens and a primary school (healthy children) or at a speech therapist's surgery or a child neurological clinic. Separated maps are trained for different types of utterances. The records contain isolated phonemes, multisyllabic words and also whole sentences. Speech parameters were calculated from the labelled data - 16 MFCC and 8 LPC, 20ms long, 50% overlap, and Hamming window. The data of healthy children are used for training and validation, while the data of children with DD are used for testing. Both part of the database are accessible for authorised users via the web. For the pilot study presented here, we use records of healthy and DD children from 6 to 10 years old from our database.

3.2 MR Diffusion-Weighted Contrast and Tractography

The magnetic resonance method is used for the quantitative description and visualization of white-matter brain structures - Arcuate Fasciculus [8]. This is a bundle of nerve fibres, myelin-insulated neurons, which connects two - relatively distant - of the most important centres of the cerebral cortex for speech processing, specifically the Broca's area in the frontal cortex (Brodmann area 44 and 45) and the Wernicke's

center in interface temporal and parietal lobe (Brodmann area 22). Using a special measuring sequence of magnetic resonance, the diffusivity of water molecules in the structures of white matter is detected, based on the assumption that the diffusion of water molecules in the white-matter structures is strongly anisotropic due to movement re-strictions in specific directions – i.e. the diffusion rate along the nerve fibres will be much greater than the diffusion of molecules in the transverse direction. The used measuring sequence determines the size of diffusion in sixteen different directions. Evaluation for a particular voxel obtains information about the overall direction of diffusion and its size. Quantitative description of the diffusion in the individual voxels is obtained by calculating parameters such as the Apparent Diffusion - Coefficient (ADC) and the Fractional Anisotropy (FA). When these parameters are known, further computer processing can reconstruct the tensor model and the probable path of the nerve fibres and obtain a visual representation of nerve bundles (called tractography). From the diffusion-weighted magnetic resonance data, the tensor model was obtained, and the subsequent method of tractography obtained an image of all neural pathways of white matter in the brain. Parameters such as Fractional Anisotropy FA, Apparent Diffusion Coefficient (ADC), length and number of threads were measured. Other parameters were determined from the final volumes. Volumes were coloured (red, left, right, blue) and reconstructed into anatomically arranged 3D structural T2-weighted TSE transversal data.

MR Tractography Analysis: The method can be described in the following steps:

1. Select all parameters / select only the promising parameters for all children.
2. Use KSOM to create 2D hexagonal map.
3. Separate data into groups using K-means in SOM toolbox [13].

The second idea is to compare the left and the right hemisphere using the amount of fibres and the volume of Arcuate Fasciculus.

4 Results

We evaluated 72 healthy children and 14 patients in treatment. Our goal was the distribution of the patients into 3 classes according to DD level (L1-mild, L2-medium, L3-severe). We compared the results from the speech analysis (from SSOM training and PCA analysis) with MR tractography results (processed by KSOM).

4.1 Classification of Children with Developmental Dysphasia Based on Speech Signal Analysis

We analyzed the vowel mapping for experiments with disordered children's speech – by SSOM and by PCA.

Classification by SSOM: SSOM was formed by a two-dimensional map contains *24 x 24* units in a hexagonal grid (in our experiment), a random initialization of the prototype vectors. Two training stages were used for our experiment:

1. The first rough stage: the Batch Map algorithm, the Gaussian neighbourhood function decreasing monotonically from 24 to 1. The number of training steps was 5000.
2. The second fine stage: the Batch Map algorithm, the Gaussian neighbourhood function decreasing monotonically from 2 to 0. The number of training steps was 1000.

The dimension of training data set was *31475 x N* (N is equal to a number of speech coefficients), the number of wav-files in the training data set was 1495, the number of phonemes in the training data set was 2299. Fig. 1 shows the results of the classification. Each figure shows the trained SSOM for vowels for the patient (2-D map on the left, U-matrix on the right). The particular clusters of vowels are represented by colours (red for \a\, orange for \e\, blue for \i\, green for \o\ and yellow for \u\).

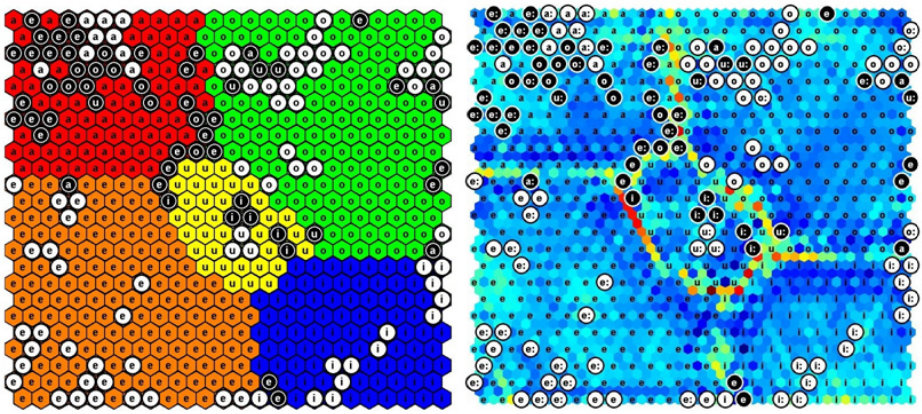


Fig. 1. SSOM maps for classification vowels from child patients speech

The training set consists of the utterances of all healthy children in the database. Utterances from a child with DD are then classified and shown within the map. White units indicate the successful classification; black units represent classification errors (wrong vowel indications are written in units). Afflicted children are not able to pronounce certain vowels or they replace them by other vowels. Characteristic replacements are \o\ for \e\ and \u\ for \i\, these replacements are typical for DD patients, but doesn't occur among healthy persons.

Classification by PCA: The value mean and variance of the "sickness" parameter among the validation sets is compared with the "sickness" parameter of each child in the testing set (children with DD). The result is the cumulative density function (CMD), in which it holds that t higher CMD, the higher degree of DD. The results are in Table 1, in column 4 and 5.

4.2 MR – Tractography Results

Four particular maps on Fig.2 show, that a volume and a number of fibres in the Arcuate Fasciculus in Fig. 2a, 2b (left hemisphere) and in Fig. 2c, 2d (right hemisphere) are different. Maps are symmetrical along the y-axis as follows from the figures. We can suppose that this fact is associated with Fig.3a, which typifies the groupings. Knowledge of this finding can help doctors to determine the degree of disorder, as this assumption is confirmed in the future for many more patients. Fig.3b is the visualization of the Arcuate Fasciculus in a child's brain.

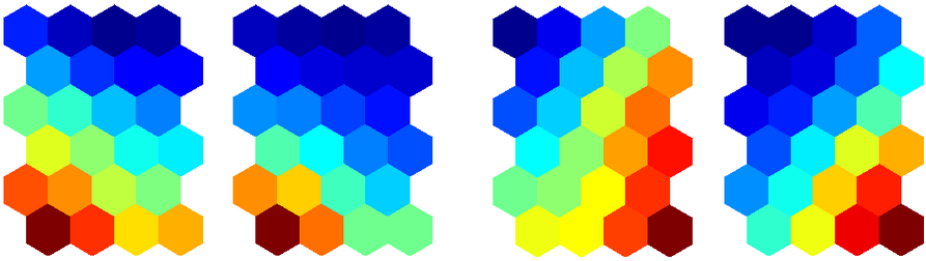


Fig. 2. SOM. The first two images are for the left hemisphere; the rest are for the right hemisphere. The first and third images represent the volume parameter and the second and fourths are the amount of fibres.

4.3 Comparison of Methods

KSOM classification to the three groups is presented in Fig. 3a. The map classifies patients into three groups based on the four selected parameters (volume and amount of fibres of both hemispheres) of MR tractography (see 3.2). The patient's labels are highlighted in the hexagonal cells. At present, it is not possible to pair groups (labelled as \A\, \B\, \C\ in Table 4 – eighth columns) with the degree of DD, because we do not have enough data to perform the correct correlations. Nevertheless, we assume that the methodology should bring us useful results, because we have good experience with KSOM in medical issues. Table 4 contains the results of MR tractography (second, third, eighth and ninth column), PCA (fourth and fifth column) and SSOM (sixth and seventh column). The second and third columns represent the ratio between maximum and minimum value (volume or amount of fibres) of the left and right hemisphere. The yellow colour means that the right hemisphere is larger and the red colour respectively the larger left hemisphere.

The results of PCA in the fourth column (explained in chapter 4.1) define the degree class of DD. The sixth column represents the correct classification by SSOM as a percentage. Based on the classification rate, we separated children into three groups using the linear method according to the results of healthy children. These values (degree of DD) will be re-valued after we obtain more data. The last two columns are related to Fig.3a (A – all data of MR tractography, B – number and volume of fibres of both hemispheres).

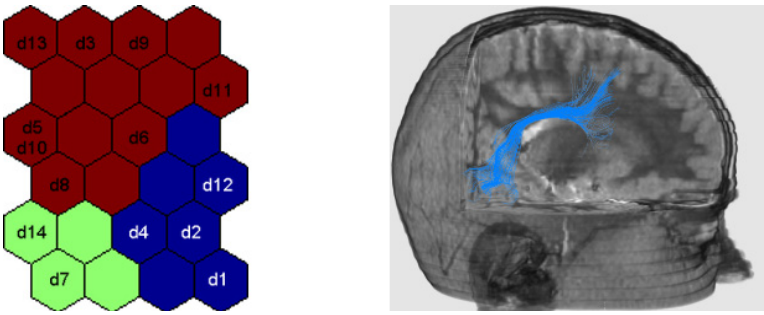


Fig. 3. The left figure represents the patient in clusters in KSOM and the right figure represents the Arcuate Fasciculus in a hemisphere

Table 4. Results

Child #	(L/R) vol	(L/R) fib	PCA CMD	PCA class	SSOM %	SSOM class	SOM MR A-GROUP	SOM MR B-GROUP
1	104,6	171,1	66,6	2	42,11	2	B	C
2	220,7	781,8	65,7	2	37,61	3	A	C
3	482,5	1220,0	71,7	3	39,69	2	A	A
4	103,6	204,9	54,7	1	39,75	2	B	C
5	128,6	178,5	76,0	3	41,10	2	B	A
6	186,7	183,3	55,4	1	56,48	1	A	A
7	158,3	195,5	53,3	1	39,14	2	B	B
8	112,3	145,4	61,7	2	38,03	3	B	A
9	610,0	1580,0	53,1	1	41,10	2	A	A
10	123,4	106,6	66,6	2	46,03	1	C	A
11	362,0	481,8	73,9	3	46,15	1	C	A
12	249,8	545,8	51,1	1	38,46	3	A	C
13	119,2	122,2	68,2	2	38,14	3	C	A
14	109,2	186,4	64,1	2	39,04	2	B	B

4.4 Software

The modular system SOMLab [10] using the MATLAB 7, Release 14 performed the analysis and some experiments. The requirement for a user-friendly graphic environment for SOM initialization and training, and for visualization of classification results, was an incentive for its creation. SAS Institute Inc. SAS/Stat® software was used for statistical computation. The SOM Toolbox was applied for creating and visualizing the SOMs. It was developed in the Laboratory CIS in the Helsinki University of Technology and is built using the MATLAB script language. The

Toolbox is available free of charge under the General Public License (GNU) from [13]. For the project, new special M-files, which should be a part of the supporting program package, were created [10]. The bilateral Arcuate fasciculus structures were manually detected, using the MedINRIA software Version 1.9.0.

5 Conclusion

We expect that the results achieved in the project will facilitate the objectivity of the disorder diagnosis process, and that a software pack will be found with a user-friendly interface for doctors or other medical staff. The current software will serve for processing of information about children with DD diagnosis and will specify the identification of logopaedic rehabilitation and its interpretation. We have suggested the use of automatic methods based on the symptoms of DD, for which the results are promising, and we assume more accurate results in a year, because we should obtain twice or even three times more extensive data of children with DD. The most promising method is SSOM, on account of the method's robustness and its visualization ability. Our aim is to find more information and correlations in the MR tractography data available at present. We assume that we can train KSOM to split data into three groups based on the data of MR tractography. In the future, we will supplement the results described in this contribution with the classifications made by psychologists and speech therapists, and with EEG results.

Acknowledgments. This study is supported by a grant of the Ministry of Health of Czech Republic, IGA MZ ČR - NT 11443-5 / 2010.

References

1. Tuckova, J., Komarek, V.: Effectiveness of speech analysis by self-organizing maps in children with developmental language disorders. Neuroendocrinology Letters 29(6), 939–948 (2008)
2. Saygin, A.P., et al.: Neural resources for processing language and environmental sounds Evidence from aphasia. Brain 126(4), 928–945 (2003), http://brain.oxfordjournals.org/
3. Moineau, S., Dronkers, N.F., Bates, E.: Exploring the processing continuum of single-word comprehension in aphasia. J. Speech Lang. Hear. Res. 48(4), 884–896 (2005)
4. de Guibert, C., et al.: Abnormal functional lateralization and activity of language brain areas in typical specific language impairment (developmental dysphasia). Brain (a Journal of Neurology) (2011), http://brain.oxfordjournals.org/ (September 19, 2012)
5. Cuingnet, R., et al.: Spatial and anatomical regularization of SVM for brain image analysis (2010), http://cogimage.dsi.cnrs.fr/perso/colliot/files/nips2010-camera_ready.pdf (September 19, 2012)
6. Kohonen, T.: Self-Organizing Maps, 3rd edn. Springer (2001)
7. Jolliffe, I.T.: Principal Component Analysis. Springer (2002) ISBN: 978-0387954424

8. Komarek, V., Kynčl, M., Šanda, J., Vránová, M.: Diffusion Tensor Imaging: Ventral and Dorsal Connections between Language Areas in Developmental Dysphasia. In: 9th EPNS Congress, Dubrovnik, Croatia (May 2011)
9. Psutka, J., Müller, L., Matousek, J., Radova, V.: Speaking with a Computer in Czech. Academia Praha (2006) (in Czech) ISBN 80-200-0203-0
10. Tuckova, J., Zetocha, P.: Speech analysis of children with developmental dysphasia by Supervised SOM. Neural Network World 16(6), 533–545 (2006) ISSN: 1210-0552
11. Vavrina, J., Zetocha, P., Tuckova, J.: Detection of degree of developmental dysphasia based on methods of vowel analysis. In: Proc. of the 36th Int. Conf. on Telecommunications and Signal Processing (TSP 2012), Prague, Czech Republic (2012) ISBN 978-1-4673-1116-8
12. Zetocha, P.: Design and realization of children speech database. Technical report, Ministry of Education grant FRVS, No.2453/2008 (2008) (in Czech)
13. Vesanto, J., Himberg, J., Alhoniemi, E., Parhankangas, J.: SOM Toolbox for Matlab. HUT (2000) ISBN 951-22-4951-0, http://www.cis.hut.fi/projects/somtoolbox

Similarity Analysis Based on Bose-Einstein Divergences for Financial Time Series

Ryszard Szupiluk[1] and Tomasz Ząbkowski[2]

[1] Warsaw School of Economics, Al. Niepodleglosci 162, 02-554 Warsaw, Poland
[2] Warsaw University of Life Sciences, Nowoursynowska 159, 02-787 Warsaw, Poland
rszupi@sgh.waw.pl, tomasz_zabkowski@sggw.pl

Abstract. Similarity assessment between financial time series is one of problems where the proper methodological choice is very important. The typical correlation approach can lead to misleading results. Often the similarity measure is opposite to the visual observations, expert's knowledge and even a common sense. The reasons of that can be associated with the properties of the correlation measure and its adequateness for analyzed data, as well as in terms of methodological aspects. In this article, we indicate disadvantages associated with the use of correlation to assess the similarity of financial time series and propose an alternative solution based on divergence measures. In particular, we focus on the Bose-Einstein divergence. The practical experiments conducted on simulated and real data confirmed our concept.

Keywords: time series similarity, divergence measures, Bose-Einstein divergence.

1 Introduction

The similarity between variables, vectors or functions can be defined in different ways. In case of the real economic problems modeling it is often expected that mathematical quantitative similarity measure corresponds to its colloquial definition. In this sense, the issue of similarity between financial instruments represented by the time series is generally recognized in two ways.

The most popular and thus the most widely explored and described are correlation methods. The similarity here is interpreted in terms of the second order statistics dependencies. This approach has a long tradition and has contributed to the solution of many fundamental issues. Its importance is not without significance for the second order statistics and the normal distribution in the data analysis. Normal distribution with its well-established statistical significance in the central statistical theorem is fully defined by the second-order statistics. Relatively simple estimation of second-order statistics, in case of Gaussian distribution, allows obtaining complete statistical information about the empirical problem. Additionally, based on the Gaussian distribution and the dominating estimation paradigm in the form of the maximum likelihood method, "comfortable" linear statistical models can be obtained [12].

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 417–427, 2013.

However, there are number of issues, where the correlation approach is not adequate. Such conditions and assumptions like non-linear dependence, data with cluster structure, nonstationarity of variables, in case of correlation analysis may lead to wrong conclusions [2,10]. In turn, transformations of variables can lead to deformation of the original relationships between variables. As a result, correlation approach requires caution and individual supervision, what makes it difficult to use in automatic pattern recognition systems [8,9]. Also on the basis of statistical methods we find some issues when the variance is by definition ambiguous. One of the examples is the use of independent component analysis (ICA). It is a multi-dimensional method addressed to non-Gaussian variables (except one possible in the set) exploring the higher-order statistics. The components obtained in this method are characterized by ambiguity with the respect to the variance. This means that it can be freely scalable, which makes that many algorithms performing ICA standardize the variances to unity [4]. At the same time, "visual" characteristics of the variability of these components can be highly different.

An alternative approach to assess the similarity can be formulated as a segmentation or clustering problem (grouping). This leads to a wide range of different techniques, in which the similarity is assessed usually with the Euclidean distance between the objects, or in the general case, the similarity is referred as the distance measured with a particular p-norm.

Let's note that it is quite popular approach, but the main drawback is the ambiguity of the obtained results. In case of two variables (time series or signals) it is difficult, based on the information about the distance between them, to assess the degree of their similarity. One solution may be to take the reference variable and to calculate the distance between these two variables and the reference signal. However, the overall similarity obtained in this way is far from ambiguity and remains relative.

These limitations were (and still are) the motivation to look for a new similarity measures. Recently, divergence measures are intensively investigated. The development of the methods based on divergences is associated due to the spectacular success of their applications, i.e. the issue of non-negative matrix factorization for images processing and patterns recognition. This opened new possibilities for pattern recognition systems used for communication with paralyzed people, by brain waves analysis [3].

One of the characteristics of divergences is the general lack of symmetry, which can be used to assess the similarity of signals (time series). In this article, we propose the method for similarity assessment in time series (signals) based on Bose-Einstein divergence. This will reveal the natural limitations of correlation analysis. The practical experiments carried out both, on simulated and real financial data confirmed the usefulness of the proposed solutions.

2 Bose-Einstein Divergence and Similarity

Divergence $D(y \| z)$, is a function of two arguments estimated on non-negative variables z i y, which satisfies the condition $D(y \| z) \geq 0$, where $D(y \| z) = 0$ if and

only if $y = z$ [1,6,7]. Divergence does not have to meet the triangle inequality $D(y \| z) \le D(y \| x) + D(x \| z)$ and the condition of the symmetry does not have to be met, too $D(y \| x) \ne D(x \| y)$. For some divergences it is needed to satisfy the condition of summation of the z and y variables to unity. Divergences may be defined for both, discrete and interval variables.

Currently, divergence functions are used to evaluate the similarity (or lack thereof) between the non-negative variables, vectors, matrices or functions. The most popular divergence classes include Bregman's and Csiszar's divergences [5,6]. The other popular divergence is Bose-Einstein divergence which, for vectors $\mathbf{z} = [z_1, z_2, ..., z_L]$ and $\mathbf{y} = [y_1, y_2, ..., y_L]$, where $y_i, z_i \in [0,1]$, is defined as [6]:

$$D_{BE}^{\alpha}(\mathbf{y} \| \mathbf{z}) = \sum_{i=1}^{L} \left(y_i \ln \frac{(1+\alpha) y_i}{y_i + \alpha z_i} + \alpha z_i \ln \frac{(1+\alpha) z_i}{y_i + \alpha z_i} \right). \tag{1}$$

where parameter $\alpha \in (0,1)$.

This divergence possesses a number of interesting properties like $D_{BE}^{\alpha}(\mathbf{y} \| \mathbf{z}) = D_{BE}^{1/\alpha}(\mathbf{y} \| \mathbf{z})$ and $D_{BE}^{\alpha \to \infty}(\mathbf{y} \| \mathbf{z}) = D_{KL}(\mathbf{y} \| \mathbf{z})$, where D_{KL} means Kullback-Leibler divergence.

The essence of divergence (1) is its lack of symmetry, which can be used to assess the similarity of signals. For the signals, which are "statistically similar" in a general sense, it is expected that the order of the arguments does not matter. In particular, for the random signals, which by definition do not have patterns or regularities, it can be assumed that the order of arguments in the asymmetric divergence (1) plays no role. This means that for noisy signals v_1, v_2 with the same distribution, the divergence measure (1) should be symmetric $D_{BE}(v_1 \| v_2) = D_{BE}(v_2 \| v_1)$. The symmetry of standardized signals [0,1], can be measured as

$$q = \mathrm{abs}\left(\log \frac{D_{BE}(z \| v)}{D_{BE}(v \| z)} \right). \tag{2}$$

By studying symmetry, using Bose-Einstein divergence, the impact of the parameter α on its value, should be presented. Note that the expression under the sign of the sum in (1) can be expressed as

$$\begin{aligned} f(y,z) &= y \ln \frac{(1+\alpha) y}{y + \alpha z} + \alpha z \ln \frac{(1+\alpha) z}{y + \alpha z} = \\ &= y \ln((1+\alpha) y) - y \ln(y + \alpha z) + \alpha z \ln((1+\alpha) z) - \alpha z \ln(y + \alpha z) \end{aligned} . \tag{3}$$

For $\alpha \in (0,1)$, all the values being logarithmized take the values of the $(0,2)$ range. In this case it is possible to expand $\ln(x)$ into Taylor series:

$$\ln x = (x-1) - \frac{(x-1)^2}{2} + ... + (-1)^{n+1} \frac{(x-1)^n}{n} + ... \approx x - 1. \tag{4}$$

As a result we obtain:

$$f(y,z) \approx y(1+\alpha)y-1)-y(y+\alpha z-1)+ \\ +\alpha z((1+\alpha)z-1)-\alpha z(y+\alpha z-1)=\alpha(y-z)^2 \quad . \tag{5}$$

Taking into account (1) and (5) for $\alpha \in (0,1)$ we have

$$D_{BE}^{\alpha}(\mathbf{y} \,\|\, \mathbf{z}) = \alpha \left\| \mathbf{y}-\mathbf{x} \right\|_2^2 + R \tag{6}$$

where R is the rest of the approximation, wherein taking into account (6) it means that R is responsible for the asymmetry $D_{BE}^{\alpha}(\mathbf{y} \,\|\, \mathbf{z})$.

This allows us to conclude that the selection of the α parameter affects sensitivity of the method for the similarity evaluation. However, in case we don't have any premises to the choice of the α parameter we can adopt $\alpha = 0.5$.

3 Self-similarity of Signals

The similarity approach based on the divergences can be used also for a single signal. For this purpose, we use a splitting technique known from rescaled range analysis (R/S) that splits the signal (time series) into parts and then measures the similarity between these fragments.

For the signal $\mathbf{x}=[x_1, x_2,...,x_N]$ with $N=nm$ observations we take m parts, each having count of n and as a result we have $\mathbf{x}=[x_1, x_2,...,x_m]$, where $\mathbf{x}_1=[x_1, x_2,...,x_n]$, $\mathbf{x}_2=[x_{n+1}, x_{n+2},...,x_{2n}]$, $\mathbf{x}_m=[x_{(m-1)n+1}, x_{(m-1)n+2},...,x_{mn}]$.

At this point, we have a set of signals among which we can calculate similarity based on divergences. That is, we can use the procedure described in paragraph 2. In order to obtain the complete information about the structure of the signal similarities we need to repeat the procedure several times dividing time series into different number of n.

Having $D_{BE}(\mathbf{x}_i \,\|\, \mathbf{x}_i)=0$, all the information about these relationships can be represented in the form of a set of matrices:

$$\mathbf{U}^{(n)} = \begin{bmatrix} 0 & D_{BE}(\mathbf{x}_1^{(n)} \,\|\, \mathbf{x}_2^{(n)}) & \cdots & D_{BE}(\mathbf{x}_1^{(n)} \,\|\, \mathbf{x}_m^{(n)}) \\ D_{BE}(\mathbf{x}_2^{(n)} \,\|\, \mathbf{x}_1^{(n)}) & 0 & \ddots & D_{BE}(\mathbf{x}_2^{(n)} \,\|\, \mathbf{x}_n^{(n)}) \\ \vdots & \vdots & \ddots & \vdots \\ D_{BE}(\mathbf{x}_n^{(n)} \,\|\, \mathbf{x}_1^{(n)}) & D_{BE}(\mathbf{x}_2^{(n)} \,\|\, \mathbf{x}_n^{(n)}) & \cdots & 0 \end{bmatrix} \tag{7}$$

then symmetry coefficients determined for the respective values of the matrix (7) have the following form:

$$q_{ij}^{(n)} = \operatorname{abs}\left(\log \frac{D_{BE}(\mathbf{x}_i^{(n)} \,\|\, \mathbf{x}_j^{(n)})}{D_{BE}(\mathbf{x}_j^{(n)} \,\|\, \mathbf{x}_i^{(n)})} \right) \tag{8}$$

while the average value of the similarity between the signal fragments having n count is:

$$\tilde{q}_t^{(n)} = \sum_{t=1}^{T} q_t^{(n)} \,. \tag{9}$$

where t runs over all indices.

The method presented above requires taking into account the number of observations when comparing the time series.

4 Practical Experiment

4.1 Areas of Application

The methods presented above can be used to assess the degree of randomness of the time series. This is important issue when choosing investment strategies on financial markets. On these markets an automated trading systems are focused on historical data analysis and patterns recognition. However, searching for patterns is reasonable in case of markets on which, indeed, these patterns are present. In terms of financial theory such markets are considered as ineffective. This means that the random walk process must take into account statistical characteristics other than those provided by the white noise model. In practice, this means testing the financial instrument (particular stocks or indices) in terms of its similarity or dissimilarity in relation to the white noise.

The possibility to assess the degree of randomness or similarity to white noise can be used also to analyze investment risk within the Value at Risk (VaR) concept. This concept involves estimation of VAR in a specific period of time. The key issue here is to estimate the probabilities of future returns. The common approach is to create a stochastic model of the financial instrument and then to conduct simulations. Here, the problem of adequate model selection usually concerns fitting it to the historical data. A part of the models based on Box-Jenkins approach requires that the signal (time series) is stationary. Testing the stationarity can be difficult to perform because the basic technique used for this purpose which is autocorrelation function testing, requires the existence of second-order statistics, and also, to some extent, it takes into account the long-term memory effects. Additionally, the approach based on autocorrelation function requires an individual assessment (introspection) of the signal characteristics and thus it is difficult to use in automatic learning.

As alternative approach to the correlation an R/S analysis with Hurst exponent interpretation can be proposed. It can be obtained by dividing signal into parts with n-observations each, calculate the variance S^2 and the range $R = \max(y) - \min(y)$ in each part what lead us to $E\{R/S\}_n = cn^H$ where c is a constant and expectation is taken over the all parts. The H value derived from regression has the following meaning. The value near 0.5 means pure white noise whereas value near 1 means the deterministic signal.

The approach to measure similarity using divergences avoids most of the problems associated with the computational aspect and the interpretation based on the autocorrelation function and the R/S analysis. It allows assessing similarity of

the time series to any reference signal, including the white noise. Additionally, it is not necessary to verify the assumptions about stationarity and ergodicity of the time series; therefore, the test can be done directly on the real prices or the return rates.

4.2 Application on Simulated Data

These considerations will be presented in the context of computer-simulated tests. The problem defined above, which is aimed at looking for a good and "intuitive", but also a quantitative measure of similarity. In other words, we test the ability to group the time series automatically, in a way that is consistent with human perception. Fig. 1 presents six signals which are independent (and thus decorrelated) with individual unity variances. It can be seen that the correlation, given in Table 1, does not correspond to the visual evaluation, in particular with regard to the last two signals (S5, S6) of the Fig.1. On the other hand, the use of distance-based p-norm $D_p\|\mathbf{x}-\mathbf{y}\|_p = \left(\sum_{i=1}^{L} | x_i - y_i |^p\right)^{1/p}$, with p = 2, leads to the results showed in Table 2.

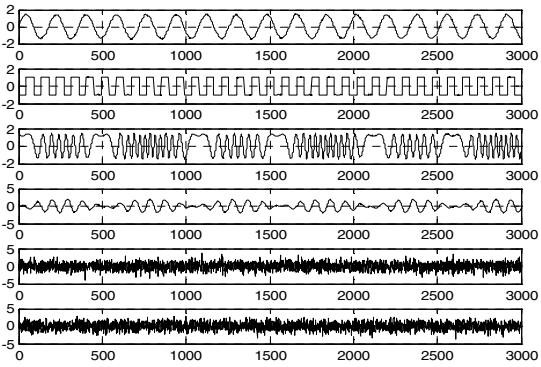


Fig. 1. Signals tested for similarity problem

Table 1. Correlation coefficients corresponding to signals presented in Fig.1

	S1	S2	S3	S4	S5	S6
S1	1.0000	0.0594	-0.0526	-0.0030	-0.0502	-0.0033
S2		1.0000	-0.0098	0.0098	0.0420	0.0142
S3			1.0000	0.0226	-0.0460	-0.0087
S4				1.0000	-0.0108	-0.0204
S5					1.0000	-0.0084
S6						1.0000

Table 2. The distances between the signals measured with p-norm, for p=2

	S1	S2	S3	S4	S5	S6
S1	0	0.0333	0.0409	0.0316	0.0287	0.0287
S2	0.0333	0	0.0379	0.0267	0.0226	0.0239
S3	0.0409	0.0379	0	0.0351	0.0331	0.0335
S4	0.0316	0.0267	0.0351	0	0.0184	0.0197
S5	0.0287	0.0226	0.0331	0.0184	0	**0.0153**
S6	0.0287	0.0239	0.0335	0.0197	**0.0153**	0

The results in Table 2 show that although the distance between the signals S5 and S6 (given in bold) is relatively accurately assessed but the symmetry of this measure significantly reduces further interpretation. It is also impossible to assess the nature of the analyzed signals.

Table 3. The distances between the signals measured with Bose-Einstein divergence for $\alpha = 0.5$

	S1	S2	S3	S4	S5	S6
S1	0	0.2354	0.4102	0.2038	0.1661	0.1674
S2	0.2415	0	0.3855	0.1525	0.1113	0.1241
S3	0.375	0.3406	0	0.2978	0.2756	0.2809
S4	0.2197	0.1641	0.3541	0	0.0663	0.0764
S5	0.1876	0.1264	0.3398	0.071	0	**0.0435**
S6	0.1881	0.1392	0.3437	0.0812	**0.0428**	0

Distance measure using Bose-Einstein divergence allows clearly differentiate similar signals what is shown in Table 3. The distances between signals S5 and S6 are given in bold and the values close to 0 indicate high similarity (accordance with the divergence definition).

In addition, the degree of symmetry for the similarity with the Bose-Einstein divergence, allows for inference about signal randomness. This is due to the fact that the noise signals have the same similarity regardless the order of the arguments used in D_{BE} divergence.

Table 4. The degree of symmetry (q) for the distances between the signals measured with Bose-Einstein divergence for $\alpha = 0.5$

	S1	S2	S3	S4	S5	S6
S1	0	0.0255	0.0898	0.0751	0.1216	0.1165
S2		0	0.1237	0.0738	0.1272	0.1147
S3			0	0.1731	0.2095	0.202
S4				0	0.0684	0.0606
S5					0	**0.0158**
S6						0

The result of the symmetry assessment to measure the similarity with parameter q (2) is presented in Table 4. The symmetry factor (q) between signals S5 and S6 is given in bold and it indicates a high similarity to noise.

4.3 Application on Financial Data

In this section we will verify the presented approach on financial data. We consider sixteen time series including stock indices and foreign exchange rates against Polish zloty, both covering the time span 04/01/2008 – 01/08/2012. Stock indices were represented by the markets of Brasil (BOVESPA), France (CAC40), Germany (DAX), London (FTSE100), Japan (Nikkei), USA (NASDAQ and SP500), Hong Kong (Hang Seng), Hungary (BUX) and Poland (WIG). Foreign exchange rates against Polish zloty included: BRLPLN, EURPLN, GBPPLN, JPYPLN, USDPLN and HUFPLN. Please, see Fig. 2 for their characteristics, Fig. 3 for their logarithmic return rates and Fig. 4 for their rescaled range analysis (R/S).

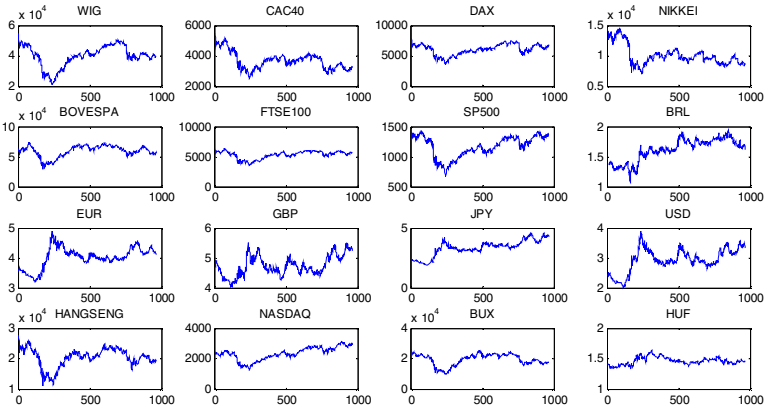


Fig. 2. Characteristics of the time series

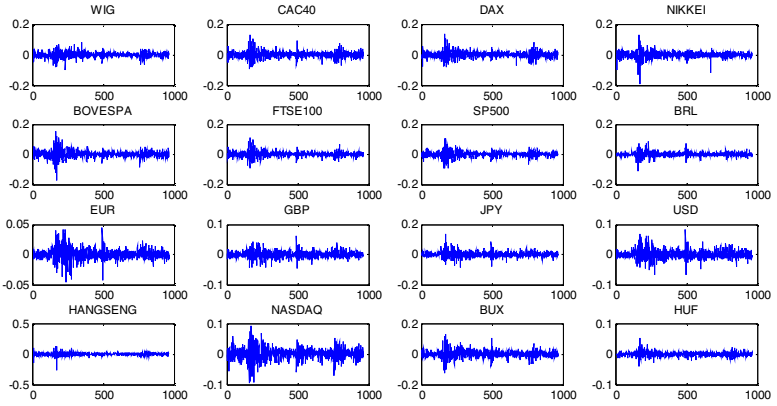


Fig. 3. Logarithmic return rates of the time series

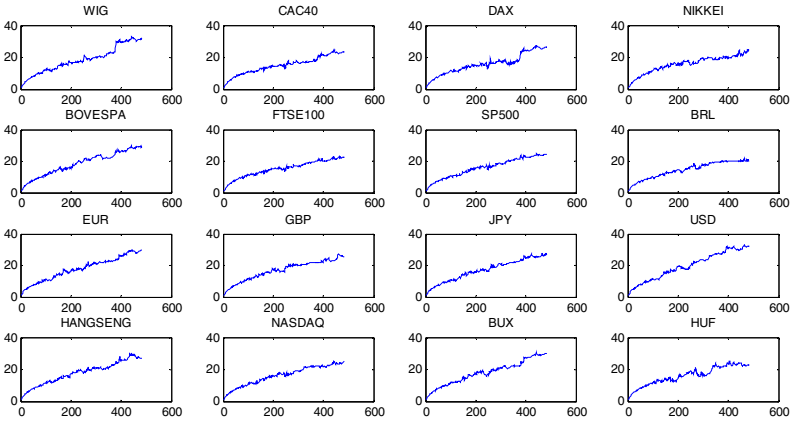


Fig. 4. Rescaled range (R/S) analysis applied to time series represented by R/S value on vertical axis and number of observations in a range on horizontal axis

For each of the time series we analyzed the degree of their randomness and then we assessed their similarity to the white noise. Table 5 presents the results of similarity assessment measured with symmetry factor (based on return rates and prices) and Hurst value (on logarithmic return rates).

Table 5. Similarity assessed by symmetry factors (q) for the return rates, for the prices and Hurst value calculated on logarithmic return rates

Instrument	Return rate q	Price q	Hurst value
WIG	55.670	78.170	0.6081
CAC40	56.194	70.351	0.5161
DAX	58.209	72.280	0.5429
Nikkei	55.192	77.096	0.5189
BOVESPA	62.018	85.953	0.6070
FTSE100	54.370	85.042	0.5153
SP500	61.392	79.640	0.5668
BRLPLN	49.280	75.962	0.5337
EURPLN	49.197	67.890	0.6084
GBPPLN	52.066	58.691	0.5704
JPYPLN	50.039	70.344	0.5913
USDPLN	47.838	73.647	0.6482
Hang Seng	59.190	79.363	0.5843
NASDAQ	53.008	88.155	0.5632
BUX	48.145	78.305	0.5957
HUFPLN	53.149	63.295	0.5298

Due to the editorial constraints the R/S approach is focused only on presenting basic characteristics in a graphical form. Based on Fig. 4, we can see that there is no evidence to differentiate or perform grouping of analyzed time series taking into account the degree of randomness. The information synthesis related to the R/S analysis and represented by the Hurst exponent, due to the number of possible regression lines fitting to the data, is neither clear.

The advantage of the divergence based approach concerns the results that are unambiguous. In addition, due to lack of assumptions regarding stochastic characteristics all the calculations can be performed both, on the return rates as well as the prices of the instruments (bearing in mind the difference in interpretation). At the same time, the results based on symmetry factor q are relatively well-interpretable in economic terms.

The lowest value (47.838), that is considered to be the most random, is currency pair USD/PLN. In this case we deal with liquid market, supposed to be the most effective, what in fact implies its similarity to random walk models. In case of the markets in which strong trends are present, such as the stock markets, the highest values of q factor (for return rates) indicate the existence of patterns in SP500 (61.392) and BOVESPA (62.018), for instance. Similar interpretation can be made for the q factor values calculated directly on instruments' prices except that the emphasis here is on the regularities and trends.

5 Conclusions

This paper presents an approach for similarity assessment using asymmetric properties of the divergence measures. Proposed Bose-Einstein divergence as a typical example of asymmetric divergence may be easily replaced by another type of divergence. Bose-Einstein divergence has interesting symmetry properties depending on the value of alpha parameter. Its changes allow differentiating the degree of asymmetry measure, which helps to adjust the sensitivity of the method depending on the data. It should be also noted that the non-negativity of the signals is not an essential restriction, since we compare and evaluate the real shape of the data sequence.

Acknowledgements. The work was funded by the National Science Center in Poland based on decision number DEC-2011/03/B/HS4/05092.

References

[1] Amari, S.: Diferential-Geometrical Methods in Statistics. Springer (1985)
[2] Anscombe, F.J.: Graphs in statistical analysis. The American Statistician 27, 17–21 (1973)
[3] Bashashati, A., Fatourechi, M., Ward, R., Birch, G.: A survey of signal processing algorithms in brain–computer interfaces based on electrical brain signals. Journal of Neural Engineering 4, 32–57 (2007)

[4] Cardoso, J.-F., Comon, P.: Independent component analysis, a survey of some algebraic methods. In: Proc. ISCAS Conference Atlanta, vol. 2, pp. 93–96 (1996)
[5] Cichocki, A., Zdunek, R., Amari, S.-i.: Csiszár's Divergences for Non-negative Matrix Factorization: Family of New Algorithms. In: Rosca, J.P., Erdogmus, D., Príncipe, J.C., Haykin, S. (eds.) ICA 2006. LNCS, vol. 3889, pp. 32–39. Springer, Heidelberg (2006)
[6] Cichocki, A., Zdunek, R., Phan, A.-H., Amari, S.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis. John Wiley (2009)
[7] Csiszar, I.: Information measures: A critical survey. In: Prague Conference on Information Theory, vol. A, pp. 73–86. Academia Prague (1974)
[8] Krutsinger, J.: Trading Systems: Secrets of the Masters. McGraw-Hill (1997)
[9] Luo, Y., Davis, D., Liu, K.: A Multi-Agent Decision Support System for Stock Trading. The IEEE Network Magazine Special Issue on Enterprise Networking and Services 16(1) (2002)
[10] Rodgers, J.L., Nicewander, W.A.: Thirteen ways to look at the correlation coefficient. The American Statistician 42(1), 59–66 (1988)
[11] Samorodnitskij, G., Taqqu, M.: Stable non-Gaussian random processes: stochastic models with infinitive variance. Chapman and Hall, New York (1994)
[12] Therrien, C.W.: Discrete Random Signals and Statistical Signal Processing. Prentice Hall, New Jersey (1992)

Exploratory Text Analysis: Data-Driven versus Human Semantic Similarity Judgments

Tiina Lindh-Knuutila and Timo Honkela

Aalto University School of Science, Department of Information and Computer Science,
P.O. Box 15400, FI-00076 AALTO, Finland
{tiina.lindh-knuutila,timo.honkela}@aalto.fi

Abstract. We present an approach for comparing human-made and automatically generated semantic representations with an assumption that neither of these has a primary status over the other. In the experimental part, we compare the results gained by using independent component analysis and the self-organizing map algorithm on word context analysis with a semantically labeled dictionary called BLESS. The data-driven methods are useful in assessing the quality of the hand-created semantic resources and these resources can be used to evaluate the outcome of the automated process. We present a number of specific findings that go beyond typical quantitative evaluations of the results of data-driven methods in which the manually created resources are usually taken as a gold standard.

1 Introduction

When the objective is to create linguistic models and theories, a traditional approach is to rely on individual linguistic intuition and knowledge building based on communication among linguists. In corpus linguistics, linguistic theories are the starting point and statistical analysis on corpus data are used to confirm, reject and refine these theories. In such a paradigm, basic linguistic categories such as noun and verb are taken as given and assumed to have a certain kind of objective status. Similarly, when computer scientists work on some linguistic data, they very often use human-constructed categories and labels as a ground truth to evaluate the performance of the computational apparatus. This kind of one-directional view on knowledge formation within computational linguistics can be seen as problematic. There is no generally accepted theory of semantics at the level of semantic categories or primitives. As a conclusion, we must consider any semantic category system or a semantically labeled corpus as a representation which may have well motivated alternatives. Based on the availability of text and speech corpora as well as sophisticated computational tools, an increasingly popular approach is data-driven: Linguistic models are created using statistical and machine learning methods.

In this article, we explore the relationship between 1) manually defined semantic categories and human judgments about whether a word belongs to a particular semantic category or another and 2) data-driven analysis of word features using unsupervised learning methods. The general architecture of this work is presented in Fig. 1. We aim to see a) whether the representations automatically generated in a data-driven manner

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 428–437, 2013.

coincide with manually constructed semantic categories, and b) critically assess manually constructed semantic categories and semantically annotated data using statistical machine learning and visualization methods.

We are particularly interested in methods that are applicable without strong linguistic assumptions. Therefore, we focus on the unsupervised learning approach rather than any supervised learning (classification) methods. More specifically, we use independent component analysis (ICA) [7] and the self-organizing map (SOM) [9]. The ICA method can be used to extract components that correspond to different categories, either syntactic [5] or semantic [12]. The SOM method is well suited for analyzing and visualizing high-dimensional data. It can show in an intuitive manner the relationships between prototypical representations of the original data points. In our application, this means that the method can visualize the relationships between different linguistic phenomena, and more specifically, between different semantic categories.

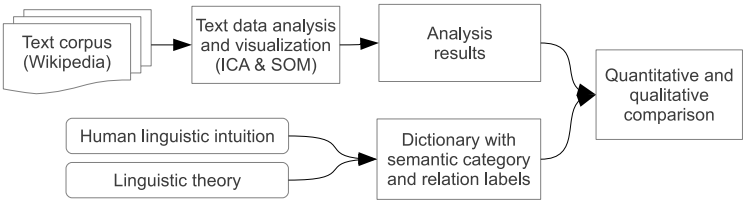


Fig. 1. The process architecture for comparing data driven and human semantic judgments

2 Semantic Similarity Judgments

In psychology, similarity judgment is considered to be one of the most central functions in human cognition (see, e.g., [4]). Humans use similarity to store and retrieve information, and to compare new situations to similar experiences in the past. Category learning and concept formation also depend on similarity judgment [18]. Research has been carried out to obtain information of human similarity judgments and different types of similarity have been identified: synonymy (*automobile:car*), antonymy (*good:bad*), hypernymy (*vehicle:car*), meronymy (*car:wheel*). A special case is a category (VEHICLE: *car, bicycle*), where the members are perceived as having some similar characteristics. Based on research in psychology and cognitive science, data sets that list words that are judged similar have been used to evaluate vector space models (see for example [1,12].), with an intuition that the similarity perceived by humans should be translated as proximity in a word vector space.

In this article, we use the BLESS (Baroni-Lenci Evaluation of Semantic Spaces) [1] test set, which is based on a body of earlier work on human similarity judgments. The data set contains 200 concepts in 17 broader classes or categories with 5-21 words per class: AMPHIBIAN_REPTILE, APPLIANCE, BIRD, BUILDING, CLOTHING, CONTAINER, FRUIT, FURNITURE, GROUND_MAMMAL, INSECT, MUSICAL_INSTRUMENT, TOOL, TREE, VEGETABLE, VEHICLE, WATER_ANIMAL, and WEAPON. Each concept is further linked with other words that are in a certain defined relation with the concept: Attributive (ATTR, describing a property of the concept), coordinating (COORD belongs to the

same category), event (EVENT, a verb related to the concept), hypernymous (HYPER, a super-ordinate concept), or meronymous (MERO, in a part-whole relation with the concept). Thus, the concept *alligator* has a COORD relation with *crocodile*, an ATTR with *carnivorous*, an EVENT with *attack*, a HYPER with *animal*, and a MERO with *eye* and *tooth*. In total, there are 14 400 word-relation pairs in the data set.

3 Unsupervised Learning in Exploratory Text Analysis

Miller and Charles [13] have presented a well-known hypothesis on the relationship between semantic similarity and context data: "two words are semantically similar to the extent that their contextual representations are similar". This basic idea is behind the corpus-based vector representations or vector space models (VSM). They capture meaning through word usage and are widely used in computational linguistics (see for example [5,11,16,17,19]). Moreover, it is assumed that relatedness equals proximity in the vector space [17]. To obtain the raw word co-occurrence count representation for N target words, the number of context words C occurring inside a window of length l positioned around each occurrence of the target words is counted. The accumulation of the context words creates a word-occurrence matrix $X_{C\times N}$. For a review on the current state of the art for vector space models, see [19]. In high-dimensional vector spaces, the most common similarity measure [19] used is the cosine similarity [11], which is also used throughout this article. The choice of the size of context around a target word is relevant when a vector space model is built. The context used can be a document, or a more immediate context around the target word. Different context sizes are systematically explored in [2]. Sahlgren [16] concludes that a small context around a target word gives rise to paradigmatic relations between words, whereas larger context allows syntagmatic relations to be more prominent. As our the concepts in categories are mostly in paradigmatic relationship, we use a bag-of-words representation with a window of length $w = 3$, i.e. the left and right adjacent word around the target word. The positive pointwise mutual information (PPMI) [14] weighting was reported to give best results [2] and is used as a weighting scheme for the VSM here.

Our corpus is built from all the documents in the English Wikipedia[1] that were over 2kB in size. In pre-processing, all wikimedia markup is removed, the words are changed into their lower case counterparts and punctuation is removed except for hyphens and apostrophes. In general VSM research, the representations are often very high-dimensional: including tens or hundreds of thousands of features. In this work, we use the 5 000 most common words as features. This choice reduces the computational load in the ICA and SOM calculation as the matrix size does not grow too large. The vocabulary contains the 200 000 most frequent words. The vector space performance has been evaluated earlier using several syntactic and semantic test sets [12]. We use a subset of the complete vector space: the words that are in the BLESS vocabulary. Within the 14 400 relations of the BLESS there are 1 673 unique word forms that appear within the 200 000 most frequent words of the Wikipedia corpus. Each word is labeled with a combination of the relation and the category, and multiple labels per word are allowed.

[1] We used the October 2008 edition, which is no longer available at the Wikipedia dump download site `http://dumps.wikimedia.org/enwiki/`

For example, a word *aeroplane* is labeled with VEHICLE-COORD and VEHICLE-HYPER, and *back* with CLOTHING-MERO, FURNITURE-MERO, MUSICAL_INSTRUMENT-MERO, and VEHICLE-MERO. We could have used even more specific labels by including the concept as well, but the number of labels would have grown too large.

The independent component analysis (ICA) [3,7] is a blind source separation method which represents a matrix of observed signals $X_{C\times N}$ as $X_{C\times N} = AS$, where $A_{C\times d}$ is a mixing matrix, and $S_{d\times N}$ contains the independent components. The columns for the matrix $S_{d\times N}$ give a d-dimensional representation for the target words. The FastICA algorithm used estimates the model by first using dimensionality reduction and whitening and then finding a rotation that maximizes the statistical independence of the components [8]. The dimensionality reduction and de-correlation step can be computed, for instance, with principal component analysis. Earlier, ICA has been used to find components that match the syntactic categories [5] and semantic categories in the Battig-Montague category test set [12]. The premise of the ICA method is that the components can be interpreted. Often the words for which the values are high in a given component are similar, which can be evaluated using known category labels.

The self-organizing map (SOM) is an unsupervised learning method which typically produces a two-dimensional discretized representation of the input space [9]. It preserves the topological properties of the input space, which makes it a useful tool for visualizing high-dimensional data. The vector space model representations are usually very high-dimensional, which make dimensionality reduction methods such as the SOM practical tools for the text data exploration (see [6,15] as examples of early works). In the SOM, the high-dimensional contexts have often been approximated with the random projection model, see e.g., [6,15]. In this article, we only use a considerably small number of the most frequent words as our context words, and thus we do not need to apply random projection.

The SOM has earlier been carefully compared with several other methods, including principal component analysis (PCA), Isomap, curvilinear component analysis (CCA), locally linear embedding (LLE) regarding their trustworthiness and continuity of the visualization. The results indicate that the SOM produces a trustworthy visualization. [21]. When the performance of the SOM algorithm is evaluated in a domain-independent manner using error resolution (quantization error) and topology preservation, these measures provide only a partial view on the goodness of the mapping in the case of a specific application such as natural language processing.

4 Finding Category Information with ICA

In this article, we compare the BLESS category and the ICA results using a similar methodology to [12]. We run ICA with 50 components for the vector representations of the 1 673 words in the BLESS vocabulary. The label information is only used in evaluation phase. We report results of one ICA run, but repeated runs yield very similar results.

In the analysis, we study the words with highest values for each independent component. As the ICA component values are usually skewed in one direction, we use the maximum values in the direction of the maximum skewness of the component. Then 10

words with largest values for each component are selected and the number of words belonging to the same category (cat), relation (rel) or joint category-relationship (cat-rel) class are calculated. Multiple labels for each word are taken into account: A word with two labels is counted once for each group. Then different thresholds are checked ($\frac{10}{10}$, $\frac{9}{10}$ etc.) to see how many categories meet each threshold condition. In [12], a minimum of $\frac{9}{10}$ words belong to the same class was defined as a *strict* threshold and a minimum of $\frac{6}{10}$ a *lax* threshold.

Fig. 2 a) shows the results for different types of categories. We can see that $\frac{33}{50}$ components describe 12 out of the 17 categories meeting at least the *lax* criterion. Similarly, $\frac{41}{50}$ components describe the five relation types. When we combine the categories and relations separately, there are $\frac{23}{85}$ different cat-rel types that meet the *lax* criterion. They cover 10 separate categories, but there are several relation types for some of the categories, sometimes even several components describe the same cat-rel type: four components for words with a GROUND MAMMAL-ATTRI label, three for VEHICLE-MERO, and two for VEHICLE-COORD and WATER_ANIMAL-COORD each. If the category types are examined separately, we notice that five categories are not covered by any component: BIRD, CONTAINER, FURNITURE, TOOL and VEGETABLE. We will return to the analysis of these categories later.

Earlier work has shown that the ICA method can be used to find components that correspond to different parts of speech [5]. Therefore, it is interesting to see how this is reflected in the results, and how especially the EVENT and ATTR relations behave. In Fig. 2 b), the total number of relations covered by the independent components are shown. The height of the bar corresponds to the total number of components, which meet the *lax* criterion for this relation type, and the gray scale stacked bars show further details. Words labeled with COORD, HYPER and MERO are mostly nouns. Our initial hypothesis was that the results would be mixed between these classes. The MERO class separated fairly well, but not necessarily within a given category. Especially the several of the animal categories have overlap. The method did not separate between COORD and HYPER, only one component describes hypernyms, but mostly they are mixed. For other classes the results are the following: four in total for EVENT and 12 for ATTR, COORD and MERO each. Nine components do not have a majority label according to this test.

The ICA results can be explored beyond checking the correspondence to the class labels. In Table 1 words with maximum values for some sample components are shown. For example, fruits are nicely found from component 18 and trees in component 32. Component 2 contains action verbs, which are correctly labeled with EVENT. In addition to those, there is for example a component for verbs example related to owning things. Earlier we noticed that several components contain attributes. Component 23 is one of them we'll look at more closely. It is difficult to draw category judgments for adjectives in a similar way to nouns, but we can note that the adjectives in this group are all short and the list contains some synonyms and antonyms: *pretty/cute-ugly* and *funny-stupid*. Other components contain different attributes: Component 7 contains materials, *(wooden, plastic, concrete...)*, whereas component 12 contains adjectives that describe behavior or status. Colors can be found in component 22 and geographical or cultural attributes in component 24. Component 18 is the only one that has a majority of hypernyms: all of them are animal category hypernyms. Some components describe

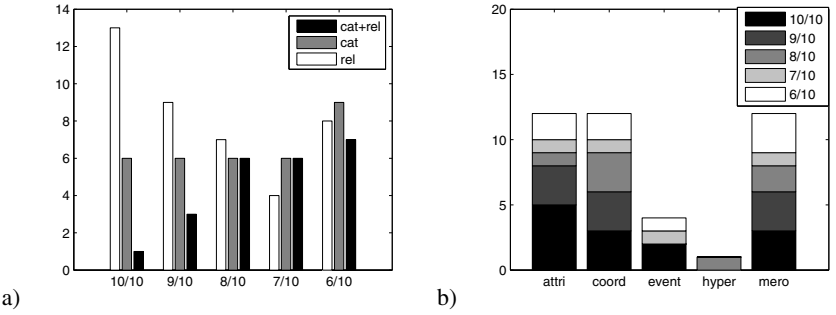


Fig. 2. a) ICA results for BLESS categories, relations and joint category-relation labels. The results indicate how many words in each component have the same category label for 10/10 to 6/10 per category type. b) A further breakdown for the five different relation types.

categories beyond predefined labels: Component 11 is an example of this. If we only look at the BLESS labels, words in this group do not seem to belong to any common category: *rock* is a FURNITURE-EVENT (to rock a chair), *pop* is an APPLIANCE-EVENT (to pop corn), and *hop* is a AMPHIBIAN_REPTILE-EVENT (a frog hops). Other words in this list, e.g,*dance, music*, show that the sense interpreted is MUSIC rather than the one they were labeled with. There were other similar examples, left out due to limited space. This phenomenon illustrates how one cannot solely rely on the predefined labels. In addition to these, we must note that in six components the words grouped together did not seem to form any meaningful category.

Table 1. Words with maximum values for sample ICA components

2	7	11	12	18	22	24	28	32
dive	steel	rock	male	strawberry	red	christian	insect	pine
jump	wooden	pop	healthy	pineapple	blue	medieval	mammal	cedar
crawl	ceramic	music	young	banana	yellow	indian	vertebrate	oak
swim	plastic	dance	solitary	citrus	white	ancient	invertebrate	cypress
kick	metal	acoustic	female	mango	purple	modern	aquatic	poplar
glide	concrete	hop	intelligent	grape	black	american	carnivorous	willow
walk	glass	jam	timid	peach	pink	religious	reptile	evergreen
climb	cardboard	metal	shy	apricot	green	asian	amphibian	birch
fly	copper	mix	faithful	watermelon	grey	african	bird	elm
float	iron	swing	peaceful	lemon	golden	roman	animal	acacia

Different senses of a word also may show in the ICA representation: a polysemous word may have several components with high values that correspond to different senses of a word. In the following, we briefly illustrate this phenomenon with a polysemous sample word *lime* (labeled in the data with FRUIT-COORD). Fig. 3 shows the ICA component representation for this word. Here, we have applied thresholding, setting all the components for which the value was less than 20% of the maximum value for that component to zero (see [20] for a more detailed description of this technique). Component

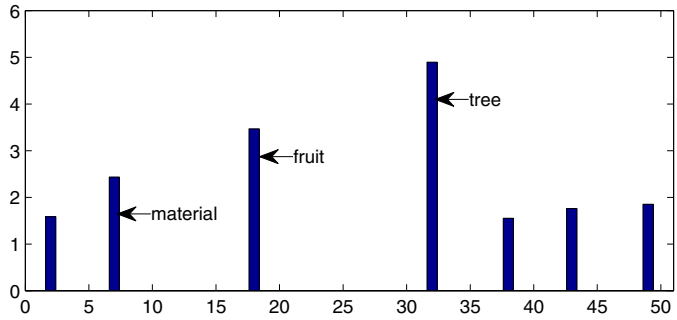


Fig. 3. Example of an ambiguous word and ICA thresholding. The components with highest values correspond to three different senses of the word *lime*. Comp. 32:TREE, Comp 18:FRUIT and Comp 7: MATERIAL.

32 has the highest value: the words in this component belong to the category TREE, so we might apply this label to *lime* as well. Component 18 has the second highest value, and indeed, words in this component have a label FRUIT. The third largest value, component 7 has a label CONTAINER-MERO. This is not enough to deduce the sense of the word, but looking at the words in this component (see Table 1) we can conclude that *lime* might be a material of some kind, hence yielding with three different senses of the word *lime*. The results are preliminary, but they suggest that the method is potentially useful in sense induction.

5 Visualization of Categories and Relations with the SOM

The self-organizing map has been used for visualizing collections of words and relationships between them [15,6]. In this paper, we use the SOM for an analysis of special cases highlighted by the ICA analysis to reveal additional structure. We trained the SOM of 200 nodes with the same word vectors of the BLESS vocabulary, using hexagonal neighborhood and a Gaussian neighborhood function. The initialization of the map is based on the largest variance of the data, according to current best practices [10]. Visualizing the complete vocabulary with the SOM and then inspecting the relations between words is easy on a computer, when the map can be examined interactively, but it is poorly suited to be presented on paper. Hence, we present here only example visualizations between categories or relations.

First, we can examine how similar the relation classes are. Fig. 4 shows hit histogram for each relation class on a grid of the trained map. The size of each dot corresponds to the number of hits in that node. A completely filled node contains five or more hits. Syntactic properties play a role in the organization of the map: verbs (EVENT) are mostly found in the the upper parts of the map, whereas nouns (COORD, MERO, HYPER) are at the bottom parts of the map, COORD mostly on right and MERO on right. Some of the ATTR words form a cluster at the middle left, but this group is quite spread out, possibly due to the fact that some of the attributes are also nouns.

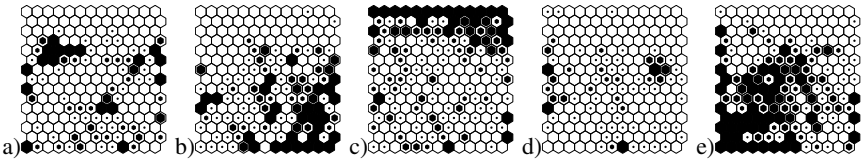


Fig. 4. Hit histograms for the five relation types. From left a) ATTR, b) COORD, c) EVENT, d) HYPER and e) MERO.

The visualization can be also used to examine the the [CATEGORY]-COORD classes (each containing 10-57 words), for example to see a) how spread out the categories are or b) which categories are overlapping or close to each other. Again, the similarity of different categories should translate into proximity on the map. Earlier, we noticed that there were five categories which were not represented by any ICA component: BIRD, CONTAINER, FURNITURE, TOOL and VEGETABLE. We can now study those categories further to find possible explanations. Fig. 5 a) shows a hit histogram for category TOOL. We can see that the there are three different centres with three hits each, and additional hits in the category have spread all over the map. This can be compared to an example of a concise hit histogram of the category TREE in Fig. 5 b). The hit histograms of the categories CONTAINER and FURNITURE is similarly scattered. The case of category the VEGETABLE in Fig. 5 c) is different. Here the category formed is fairly concise, with two separate centers. Here we compare it to a related category, FRUIT, shown in Fig. 5 d). This analysis shows that these categories overlap, and indeed, the independent component that mostly contains words in the FRUIT class covers some VEGETABLE category words as well. In a similar way, the category BIRD overlaps with other animal categories.

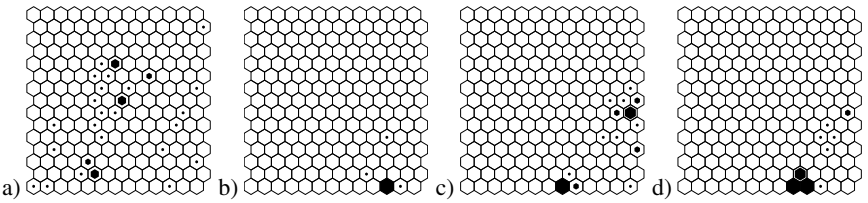


Fig. 5. Examples of category visualization with the SOM. In TOOL, most instances of the category have spread out on the map Compare with a concise category TREE (b). The pair VEGETABLE (c) and FRUIT (d) are an example of overlapping categories.

6 Conclusions and Discussion

We have compared the structure found with independent component analysis to semantic labels of the BLESS semantic dictionary, and found out that ICA is able to find components that are semantically interpretable. ICA cannot find all manually defined

class labels in an unsupervised way. Instead, it finds structures from the data that may or may not correspond to class labels. Comparison of the automatically generated structures and manually defined classes provides useful information. In order to explore this relationship in more detail, we have demonstrated how the SOM can be used for this purpose. It serves as a visualization tool for category information, which can yield information on the conciseness of the categories or relations between different categories. This work can be further extended by combining different separate data for more labeled data or comparing the ICA results with other manually built resources such as ontologies. In addition, the choice of the number of independent components with regard to the size of the data set could be inspected in more detail.

Acknowledgement. T. L.-K. has been supported by the Finnish Cultural Foundation. T.H. wishes to acknowledge the META-NET Network of Excellence for the useful discussions. The authors thank Jaakko J. Väyrynen for his valuable help and the anonymous reviewers for their comments.

References

1. Baroni, M., Lenci, A.: How we blessed distributional semantic evaluation. In: Pado, S., Peirsman, Y. (eds.) Proc. of EMNLP 2012, Geometrical Models for Natural Language Semantics (GEMS 2011) Workshop, pp. 1–10. ACL, East Stroudsburg (2011)
2. Bullinaria, J., Levy, J.: Extracting semantic representations from word co-occurrence statistics: A computational study. Behavior Research Methods 39, 510–526 (2007)
3. Comon, P.: Independent component analysis—a new concept? Signal Processing 36, 287–314 (1994)
4. Goldstone, R.: The role of similarity in categorization: Providing a groundwork. Cognition 52, 125–157 (1994)
5. Honkela, T., Hyvärinen, A., Väyrynen, J.: WordICA — emergence of linguistic representations for words by independent component analysis. Natural Language Engineering 16, 277–308 (2010)
6. Honkela, T., Pulkki, V., Kohonen, T.: Contextual relations of words in Grimm tales, analyzed by self-organizing map. In: Fogelman-Soulié, F., Gallinari, P. (eds.) Proc. of ICANN 1995, pp. 3–7. EC2, Nanterre (1995)
7. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley & Sons (2001)
8. Hyvärinen, A., Oja, E.: A fast fixed-point algorithm for independent component analysis. Neural Computation 9(7), 1483–1492 (1997)
9. Kohonen, T.: Self-Organizing maps. Springer, Heidelberg (2001)
10. Kohonen, T., Honkela, T.: Kohonen network. Scholarpedia 2(1), 1568 (2007)
11. Landauer, T., Dumais, S.: A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychological Review 104(2), 211–240 (1997)
12. Lindh-Knuutila, T., Väyrynen, J., Honkela, T.: Semantic analysis in word vector spaces with ICA and feature selection. In: Jancsary, J. (ed.) Proc. of The 11th Conference on Natural Language Processing (KONVENS), pp. 98–107. ÖGAI (2012)
13. Miller, G., Charles, W.: Contextual correlates of semantic similarity. Language and Cognitive Processes pp. 1–28 (1991)

14. Niwa, Y., Nitta, Y.: Co-occurrence vectors from corpora vs. distance vectors from dictionaries. In: Proc. of COLING 1994, pp. 304–309 (1994)
15. Ritter, H., Kohonen, T.: Self-organizing semantic maps. Biological Cybernetics 61, 241–254 (1989)
16. Sahlgren, M.: The word-space model: using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces. Ph.D. thesis, Stocholm University, Department of Linguistics (2006)
17. Schütze, H.: Word space. In: Advances in Neural Information Processing Systems, vol. 5, pp. 895–902. Morgan Kaufmann (1993)
18. Schwering, A.: Approaches to semantic similarity measurement for geo-spatial data: A survey. Transactions in GIS 12(1), 5–29 (2008)
19. Turney, P., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of Artificial Intelligence Research 37, 141–188 (2010)
20. Väyrynen, J.J., Lindqvist, L., Honkela, T.: Sparse distributed representations for words with thresholded independent component analysis. In: Si, J., Sun, R. (eds.) Proc. of IJCNN 2007, pp. 1031–1036. IEEE (2007)
21. Venna, J., Kaski, S.: Local multidimensional scaling. Neural Networks 19(6), 889–899 (2006)

Linear Support Vector Machines for Error Correction in Optical Data Transmission

Alex Metaxas[1], Alexei Redyuk[2], Yi Sun[1], Alex Shafarenko[1], Neil Davey[1], and Rod Adams[1]

[1] Biological and Neural Computation Research Group, School of Computer Science University of Hertfordshire, Hatfield, Herts. AL10 9AB UK
{a.metaxas,y.2.sun,a.shafarenko,n.davey,r.g.adams}@herts.ac.uk
http://homepages.feis.herts.ac.uk/~nngroup/

[2] Novosibirsk State University, Novosibirsk, 2 Pirogova street, 630090, Russia
alexey.redyuk@gmail.com

Abstract. Reduction of bit error rates in optical transmission systems is an important task that is difficult to achieve. As speeds increase, the difficulty in reducing bit error rates also increases. Channels have differing characteristics, which may change over time, and any error correction employed must be capable of operating at extremely high speeds. In this paper, a linear support vector machine is used to classify large-scale data sets of simulated optical transmission data in order to demonstrate their effectiveness at reducing bit error rates and their adaptability to the specifics of each channel. For the classification, LIBLINEAR is used, which is related to the popular LIBSVM classifier. It is found that it is possible to reduce the error rate on a very noisy channel to about 3 bits in a thousand. This is done by a linear separator that can be built in hardware and can operate at the high speed required of an operationally useful decoder.

Keywords: Error correction, classification, optical communication, adaptive signal processing.

1 Introduction

Fibre optic communication links are extensively used for high-speed and long-distance data transmission [7]. For example, the internet backbone primarily consists of fibre optics trunk lines, bundles of fibre optic cables combined together to provide increased capacity (e.g. Trans-Atlantic links). Furthermore, Nielsen's Law of Internet bandwidth [9] states that "a high end user's connection speed grows by 50% per year", an exponential growth of bandwidth year on year. Dutton highlights the problem in [4]: The faster the link the lower we need the error rate to be! But the harder that low error rate becomes to deliver. Therefore, improving the performance (lowering the *Bit Error Rate*, *BER*) of fibre optic links is not only an important task but is one that is also difficult

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 438–445, 2013.

to achieve. Fibre optic link performance is affected by a variety of phenomena, described in the next section, which may combine to cause signal degradation. In addition, each particular link has its own characteristic signature of transmission impairments [10] [6]. As stated by Hunt et al in [7]: There is great value in a signal post-processing system that can undo some of these signal distortions, or that can separate line-specific distortions from non-recoverable errors. Signal post-processing in optical data communication can offer new margins in system performance in addition to other enabling techniques.

In this paper we build on our earlier work by using a much bigger and noisier data set than we have previously analysed. In order to work with such a data set we have used an optimised linear SVM which improves upon our previous use of a neural network approach using a pereptron based method.

2 Background

Communication of digital signals along physical media typically requires that the bits are encoded into a time-varying signal at the transmitter, transmitted along the medium, and then decoded back into a digital signal at the receiver. The basic operation of an optical communication system is as follows (see Figure 1)[4]: A serial bit stream in electrical form is presented to a modulator, which encodes the data appropriately for fibre transmission. A light source (laser or Light Emitting Diode - *LED*) is driven by the modulator and the light focused into the fibre. The light travels down the fibre (during which time it may experience dispersion and loss of strength). At the receiver end the light is fed to a detector and converted to electrical form. The signal is then amplified and fed to another detector, which isolates the individual state changes and their timing. It then decodes the sequence of state changes and reconstructs the original bit stream. The timed bit stream so received may then be fed to a using device.

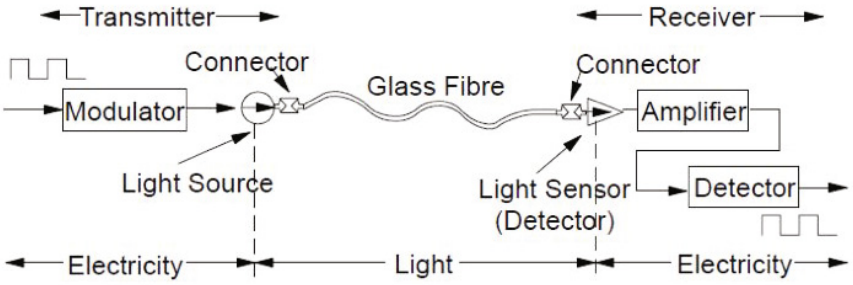


Fig. 1. The process of optical transmission

There are three broad categories of signal degradation in optical systems [1]: Attenuation – decay of signal strength, or loss of light power, as the signal propagates through the fiber. Chromatic dispersion – spreading of light pulses as they travel down the fiber. Nonlinear effects – cumulative effects from the interaction of light with the material through which it travels, resulting in changes in the light wave and interactions between the light waves.

In this paper we attempt to use a trainable classifier to help reduce the number of bits that are incorrectly decoded due to degradation.

One important feature of this problem domain is that if the resulting trained classifier is going to be useful it must be extremely fast. Optical channels can operate at speeds of over 50GHz. Clearly a classifier will only be useful if it is built in hardware. To this end we have used a simple linear separator, which can easily be built in hardware. In a related application this speed requirement is discussed and an SVM is instantiated on a *FPGA* (Field-Programmable Gate Array) board and classification is done at over 10GHz (see [8]). In our earlier work we found the linear separator using perceptron learning in a neraul network abased approach. However training a perceptron to find a good separator, particularly on a large data set is known to be difficult [3]. So in this work we use a linear SVM to find an effective linear separator.

3 The Data

The data is of the form described in [7], consisting of a large number of bits encoded as the electrical signal produced following the conversion of the optical signal into an electrical current. Each bit is therefore encoded as a waveform. The waveform is represented by using 32 evenly spaced samples of the intensity level within a bit time slot, producing a 32-ary vector of real numbers. The sum of these 32 values is the *energy* of the signal. Figure 2 shows an example of a stream of five bits. The original bit stream is also recorded so that each wave has an associated binary label.

The data was produced by a simulation of a single transmission channel, which was deliberately made to have a high level of noise, in order to produce misclassifications. So the data set we use is large, consisting of a sequence of 611,430 bits of which 105,890 or 17.32% are misclassified by an optimal energy thresholder. We divided this data set into 4/5 training and 1/5 testing, by using the last 122,286 bits as the test set. For the whole data set we searched for the energy threshold (a value below/above which a wave is decoded as a 0/1) that gave the best decoding of the data stream, that is it gave the best reconstruction of the original binary data stream. As this is a very noisy channel the error rate even with the best threshold is high. We denote those bits in the data stream that are correctly decoded by the threshold as *easy*, and those that are incorrectly decoded as *hard.* Table 1 gives a breakdown of the data set.

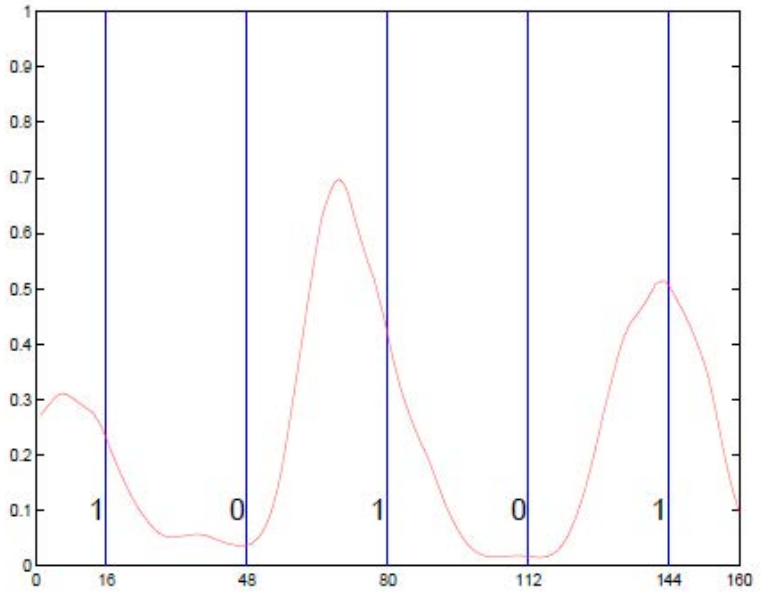


Fig. 2. An Example bit stream

Table 1. The breakdown of the data set

Data Set	*Training*	*Testing*
Easy	404,432	101,108
Hard	84,712	21,178
Total	489,144	122,286
Easy(%)	82.68	
Hard(%)	17.32	

The hard bits usually come from either the sequence "101" or "010" where the central bit is often distorted by the energy of the bits surrounding it. Figure 3 shows some examples of the "010" subsequence. It can be seen that the red misclassifications do not have sufficient energy to be classified as 1's.

In order to represent this data for a trainable classifier we simple took the 32-ary wave vector for each bit and catenated the representation of the bits to its left and right, giving a 96-ary real vector. The motivation for this was that the surrounding bits have a clear influence on the wave of the bit between them and this information could be of use to the classifier. In summary our data set consists of 611,430 96-ary labelled real valued vectors.

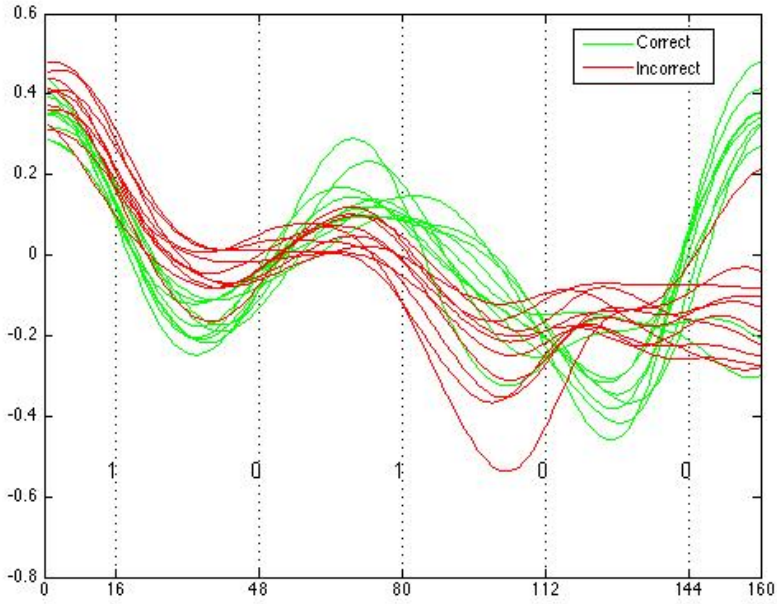


Fig. 3. Waves where the central bit is correctly or incorrectly decoded by the threshold

4 The Classifier

As we have said earlier we use a linear SVM to find a good separator of our data.

4.1 Software Used

The actual tool we used is LIBLINEAR [5] which is a linear classifier produced by the authors of the well known LIBSVM [2]. It supports the same data formats as LIBSVM but is more suited to classification of large data sets with [5]: "millions of instances and features".

4.2 Training

The only hyper parameter in a linear SVM is the regularising cost parameter C. To find a good value for C we simply undertook an empirical search using 5 fold cross validation in the training set.

5 Results

The first thing to note is that LIBLINEAR handled this huge data set without difficulty. This is quite impressive as the training set alone contained 489,144

96-ary vectors, or 46,957,824 real numbers. The search for a good value for C took about an hour on an Intel QX6700 Core 2 Extreme processor. Table 2 gives the final classification rates on the test set.data in order to demonstrate their effectiveness at reducing bit error. The error split for the Thresholder is not given since by definition the easy set are the bits correctly thresholded and the hard set those incorrectly thresholded.

Table 2. Final Results

Classifier	*Accuracy (%)*	*Error Rate (%)*	*Error Split*	
Threshold	82.68	17.32	*easy set (%)*	*hard set (%)*
LIBLINEAR	99.62	0.38	62.7	37.3

We can see that the SVM has corrected many of the original errors. In numerical terms the 21,178 original errors have been reduced to just 437. We cannot make a direct comparison with our earlier work, using perceptrons, as we have never before used such a large data set. However on a subset of this data, about one fifth, we were previously able to get a best error rate of 1.15% [6], as against 0.38% here.

Figure 4 shows a wave that the SVM was able to correct and one that it could not correct. For example it is able to correct the red wave (B) which has poor alignment but in context is recognisably a "one". However the magenta wave (C) has both poor alignment and poor shape and the classifier is unable to correct it.

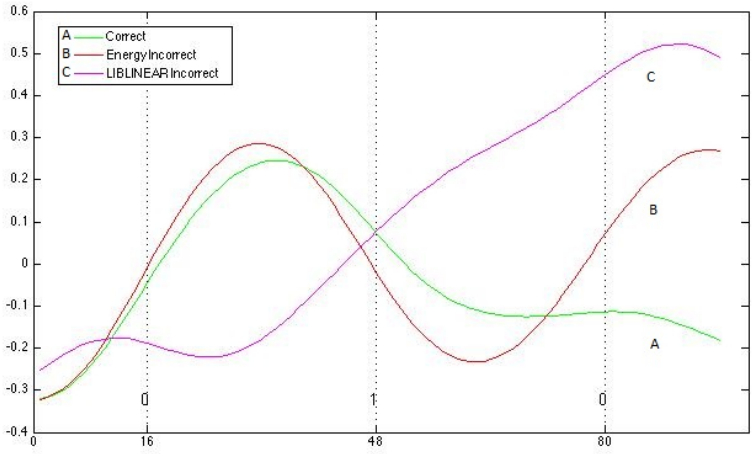


Fig. 4. The Classification of various waves

Table 3 shows the number of incorrectly identified 3 bit sequences (in which the middle bit is incorrectly identified). Notice that, as is usually the case it is the "101" and '010" sequences that present the biggest problem for both thresholder and the SVM. Nevertheless the classifier is able to correct many of the errors of the thresholder. For example the 6,411 thresholder decoding errors for the "010" sequence are reduced to just 128 by the SVM. The thresholder makes no misclassifications of the "000" bit sequences. It does make a small number misclassification of "011" and "100". Notice that the SVM does a little worse than the thresholder for these three cases, presumably so as not to overfit the training data so that many of the other errors can be corrected.

Table 3. Number of Errors Made

Sequence	Threshold Errors	SVM Errors
000	0	1
001	4,102	35
010	6,411	128
011	19	34
100	16	26
101	6,439	179
110	3,987	20
111	138	14

6 Discussion

6.1 Analysis of Results

The results of the paper show, quite definitively, that error correction of optical signals using linear support vector machines can approach the target BER (as stated in [10]) of 0.1%, or less than one erroneous bit in a 1000. This is true even in the case of a very noisy channel with high thresholded BER, as demonstrated by the massive reduction in error produced by the SVM. In the experiments the linear kernel SVM (LIBLINEAR) achieved significant gains over the previously achieved results using neural networks and other trainable classifiers. Importantly it is possible to build a hardware based classifier that can work at speeds of over 10GHz, and by parallelising the classification in an appropriate way speeds of over 100GHz should be possible. Moreover a FPGA board based classifier can be reprogrammed should the characteristics of the data channel, being decoded, change.

6.2 Linear Kernel SVM

The performance of LIBLINEAR was notable due to both its improvement over the previous results in [6], a 70% reduction of the BER, and its high operating

speed. It by far outstripped the training and prediction speeds of other classifiers we have used, making it possible to analyse the very large data set presented here. This indicates that it may be more easy to implement in hardware and certainly that, due to its reduced dimensionality, its computational cost is low. While it didn't achieve the target BER of 0.1% it is certainly worth further investigation. In other work [10] we have used different representations of the wave, for example adding the energy of the waves to the input and it would be interesting to try this out with LIBLINEAR. Also, a more extensive search of the C space may locate more optimal settings as a number of local minima were observed.

References

1. Bernstein, G., Rajagopalan, B., Saha, D.: Optical Network Control: Architecture, Protocols, and Standards. Addison Wesley (August 2003)
2. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines (2001)
3. Collobert, R., Bengio, S.: Links between perceptrons, mlps and svms. In: Brodley, C.E. (ed.) ICML. ACM International Conference Proceeding Series, vol. 69. ACM (2004)
4. Harry, J.R.: Dutton. Understanding optical communications. Prentice Hall PTR (1998)
5. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J.: LIBLINEAR: a library for large linear classification. The Journal of Machine Learning Research 9, 1871–1874 (2008)
6. Hunt, S., Sun, Y., Shafarenko, A., Adams, R., Davey, N., Slater, B., Bhamber, R., Boscolo, S., Turitsyn, S.K.: Correcting Errors in Optical Data Transmission Using Neural Networks. In: Diamantaras, K., Duch, W., Iliadis, L.S. (eds.) ICANN 2010, Part II. LNCS, vol. 6353, pp. 448–457. Springer, Heidelberg (2010)
7. Hunt, S., Sun, Y., Shafarenko, A., Adams, R., Davey, N., Slater, B., Bhamber, R., Boscolo, S., Turitsyn, S.K.: Adaptive Electrical Signal Post-processing with Varying Representations in Optical Communication Systems. In: Palmer-Brown, D., Draganova, C., Pimenidis, E., Mouratidis, H. (eds.) EANN 2009. CCIS, vol. 43, pp. 235–245. Springer, Heidelberg (2009)
8. Maliuk, D., Stratigopoulos, H.-G., Makris, Y.: An analog vlsi multilayer perceptron and its application towards built-in self-test in analog circuits. In: Proceedings of the 2010 IEEE 16th International On-Line Testing Symposium, IOLTS 2010, pp. 71–76. IEEE Computer Society, Washington, DC (2010)
9. Nielsen, J.: Nielsen's law of internet bandwidth (1998), `http://www.useit.com/alertbox/980405.html`
10. Sun, Y., Shafarenko, A., Adams, R., Davey, N., Slater, B., Bhamber, R., Boscolo, S., Turitsyn, S.K.: Adaptive electrical signal Post-Processing in optical communication systems (2008)

Windows of Driver Gaze Data: How Early and How Much for Robust Predictions of Driver Intent?

Firas Lethaus[1,⋆], Rachel M. Harris[2], Martin R.K. Baumann[1], Frank Köster[1], and Karsten Lemmer[1]

[1] Institute of Transportation Systems,
German Aerospace Center (DLR), Braunschweig, Germany
{firas.lethaus,martin.baumann,frank.koester,karsten.lemmer}@dlr.de
[2] Department of Electronic and Electrical Engineering,
University of Strathclyde, Glasgow, United Kingdom
rachel.harris@gmx.net

Abstract. Previous work has demonstrated that distinct gaze patterns precede certain driving manoeuvres [1,2] and that they can be used to build an artificial neural network model which predicts a driver's intended manoeuvres [3,4]. This study seeks to move closer towards the goal of using gaze data in Advanced Driver Assistance Systems (ADAS) so that they can correctly infer the intentions of the driver from what is implied by the available incoming data. Drivers' gaze behaviour was measured in a dynamic driving simulator. The amount of gaze data required to make predictions that manoeuvres will occur and the reliablity of these predictions at increasing pre-manoeuvre times were investigated by using various sized windows of gaze data. The relative difficulty of predicting different manoeuvres and the accuracy of the models at different pre-manoeuvre times are discussed.

Keywords: Advanced Driver Assistance Systems, Driver Inference, Driver Intent, Gaze Patterns, Manoeuvre Recognition, Artificial Neural Networks.

1 Introduction

Advanced Driver Assistance Systems (ADAS) must deliver correct and timely assistance to the driver. False alarms and any subsequent intervention or the correct assistance delivered too late can cause additional driving errors to be made, for example, the case of an intended overtaking manoeuvre being mistaken for an impending collision with the car to be overtaken. An ADAS is therefore required to make robust predictions of the drivers' intentions. The problem considered here is that of predicting lane change manoeuvres. A Markov chain analysis of driver gaze data demonstrated that there are distinct patterns of gaze prior to

⋆ Corresponding author.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 446–455, 2013.

the onset of specific manoeuvres [1,2]. Using the fact that gaze behaviour is an indicator of information gathering [5], it was demonstrated previously that driver gaze data can be used as a stand-alone source of training and test data when producing artificial neural network models (ANN) which predict the occurance of driving manoeuvres [4]. These ANNs thereby inferred the drivers' intent from the gaze data. In this study as with the previous study [4], one task was to predict whether the driver would change lane left or keep in the lane, the other task was to predict whether the driver would change lane right or keep in the lane. Here, the ANNs were trained using a moving window of gaze data which was varied in size from 0.5 seconds to 10.0 seconds and was positioned at increasing time from the onset of the manoeuvre in order to ascertain the effect on the ability of the ANN models to predict the occurance of a manoeuvre. The cockpit of the car visible to the driver was divided into five non-overlapping viewing zones described in section 3.3. For each instance of a manoeuvre to be predicted, a 10 second section of driver gaze data was available prior to its commencement. In the previous study [4,3], the results of using only the 5 seconds of data prior to the manoeuvre occuring and the full 10 seconds of data were compared for pre-manoeuvre times ranging from 0.0 seconds to 2.0 seconds. In the study presented here, windows of the 10 second gaze data samples formed the input vectors used to train and test the ANN models. These windows ranged in size from 0.5 seconds to 5 seconds increasing in 0.5 second increments. The range of pre-manoeuvre times investigated was restricted by the data window in use, ranging from 0 seconds to $(10 - window)$ seconds. This allowed the effect upon predictive ability to be explored in terms of the amount of gaze data used at different pre-manoeuvre times.

2 Recognition of Driving Manoeuvres

It was noted that Driver Intent Inference (DII) is distinct from Trajectory Forecasting (TF) approaches [6]. A DII approach infers if / when a driver is intentionally about to execute a lane change whereas a TF approach predicts whether the vehicle trajectory is likely to cross the lane boundary in the near future (irrespective of driver awareness level). It was also stated in [6] that most other approaches perform TF using the results as a proxy for DII. This can be summarised by stating that the absence of data derived from measurement of the driver means that nothing can be inferred about the driver's intent.

A number of studies have been carried out in real traffic as well as in driving simulators, focussing on recognising and identifying driving manoeuvres by incorporating driver data. Sparse Bayesian Learning was used to predict lane change manoeuvres when using car data alone and when incorporating driver state information in the form of head movement data [6]. It was found that the inclusion of this driver state information resulted in the predictions of lane change manoeuvre at 3.0 seconds before manoeuvre that were as accurate as the predictions made at 2.5 seconds before manoeuvre using car data only. Hidden Markov Models (HMMs) were applied to vehicle data from real traffic in

order to recognise and identify driving manoeuvres using a batch algorithm [7]. Contextual information, such as gaze behaviour, lane, and surrounding traffic were used as inputs to the models. Gaze was fed into the model as a discrete signal with six possible values (front road, rear view mirror, right mirror, left mirror, right, and left). Drivers' gaze behaviour was identified to be a relevant feature for driving manoeuvre prediction and recognition, predominantly in connection with lane changes, overtaking, and executing turns. A combination of vehicle and gaze data delivered the best results. Further results showed that the discrimination of manoeuvres, such as Overtaking and Lane Change Left, is relatively poor, if only based on vehicle data, and that recognising turns and lane changes requires contextual information. On average, driving manoeuvres were recognised one second prior to a significant change. ANN models were trained to predict the onset of Lane Change Left (LCL) and Lane Change Right (LCR) manoeuvres using driver gaze data alone [3,4]. Both 10 second and 5 second sections of pre-manoeuvre gaze data were used to investigate predictive ability at times before manoeuvre ranging from 0 to 2 seconds before manoeuvre onset. Predictive ability well above that obtained from random guessing was achieved for all times before manoeuvre onset that were trialled.

These previous studies did not investigate the limit at which any degree of prediction can still be seen to take place, which is important if future work is to be focussed correctly on where to improve. This study aims to find the limits of predictive ability with the models and data available.

3 Simulator Study

The driving task carried out by each driver took place in a dynamic driving simulator allowing the volume of traffic to be controlled, which is a known problem when conducting real-world studies. The use of a simulated environment also ensured that all drivers were exposed to the same driving conditions, that many safe opportunities to change lane could be created and that the scenarios the drivers were presented with could be safely repeated.

3.1 Driving Task

The study included a total of ten participants (five female, five male) aged 23 to 36 years (M=29.8, SD=4.6). All had normal vision, had held their driving licence for at least 5 years and drove more than 10,000 kilometres p.a. (∼6250 miles p.a.). Informed consent was obtained from each driver who participated prior to testing. The driving task took place in simulated traffic and comprised a drive on a three-lane and two-lane motorway each having a length of 70 kilometres (∼43.5 miles). Drivers were instructed to drive on the right-most lane throughout the experiment and to only use the centre lane (for three-lane motorway) or the left-most lane (two-lane motorway) for the purpose of overtaking lead cars. Each drive took approximately 40 minutes and started with overtaking a group of lead cars, which was repeated ten times, was followed by 10 kilometres (∼6.2

miles) of car following, and ended with overtaking a single lead car, also repeated ten times. Prior to the beginning of the experiment, drivers were given oral and written instructions, followed by a gaze calibration procedure of the eye tracking system.

3.2 Equipment

The driving task was performed in a dynamic driving simulator, a motion system based on a hexapod system, which allows motion with six degrees of freedom and which is illustrated in Fig. 1 and described in detail in [4]. Eye movements were recorded using a head-mounted eye tracking system, SMI iView X^{TM} HED, and five non-overlapping viewing zones were defined inside the vehicle (windscreen, left window/wing mirror, rear view mirror, speedometer, right window/wing mirror) in order to analyse the driver's gaze behaviour. All data was recorded at a sampling rate of 30Hz.

Fig. 1. Dynamic driving simulator at the German Aerospace Center

3.3 Data Processing

The section of the car cockpit that was visible to the driver was divided into five viewing zones as illustrated in Fig. 2 (1 = windscreen, 2 = left window/wing mirror, 3 = rear-view mirror, 4 = speedometer, 5 = right window/wing mirror). For each driver, the gaze data collected formed a time series of the zones looked into as the driving task was performed. The data gathered from each driver was analysed and the point at which any of the manoeuvres of interest was found to begin was noted. For each of these manoeuvre events, the 10 second section of data which preceded it was stored.

Previous work [1,2] has shown that the 10 second window of data preceding the manoeuvre was rich enough in information that distinct gaze patterns could be recognised. Redundancy within this 10 second sample length was investigated

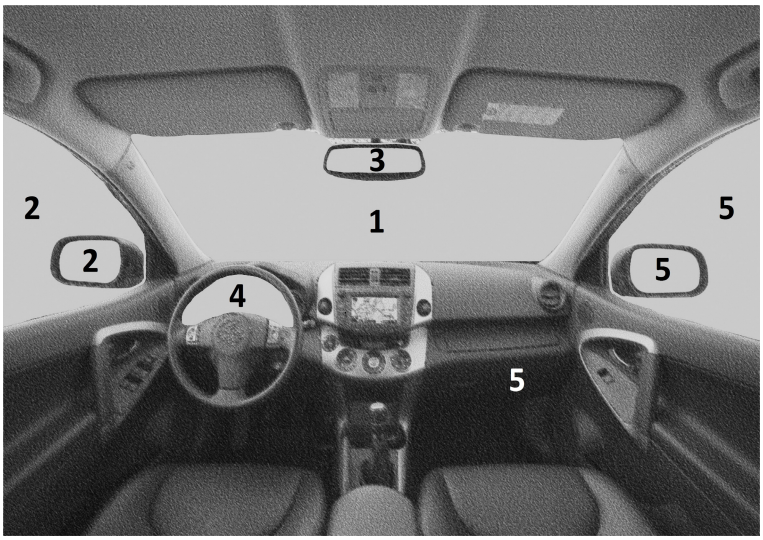


Fig. 2. Illustration of the locations of the five viewing zones inside the vehicle cockpit

further in [4] where it was shown that better predictive ANN models could be trained using data only from the 5 seconds preceding the onset of the manoeuvres. It was concluded that the additional information in the full 10 seconds of data constituted noise. The aim here was to establish the limits of the data in producing predictive models which provide a result better than chance, specifically: How much data is needed? How far in advance of the manoeuvre's onset can predictions be made? This was achieved by training ANNs using moving windows of data selected from the 10 second sample available prior to the manoeuvre onset. ANNs were trained using windows of data ranging in size from 0.5 seconds to 5 seconds with the window size increasing in 0.5 second increments. Each window of data was moved from 0.0 seconds before the manoeuvre to $(10 - window)$ seconds before the manoeuvre, e.g. a 0.5 second window would take positions 0.0 seconds before manoeuvre to 9.5 seconds before manoeuvre moving in 0.5 second increments. For each window size and for each time before manoeuvre (TBM) onset the test error rate reported was obtained by dividing the data set into 10 equal portions and training and testing as per 10-fold cross-validation. For each fold, the train and test procedure was repeated 10 times, each time using a ANN which had a different set of randomly assigned weights, so that the effect of the initial weights could be ameliorated (see section 5). The distribution of gaze behaviour across all five viewing zones from a 10 second sample of data prior to a LCL manoeuvre is shown in Fig. 3 where a 3 second window positioned at 2.5 seconds before manoeuvre onset has been overlayed onto this gaze data series.

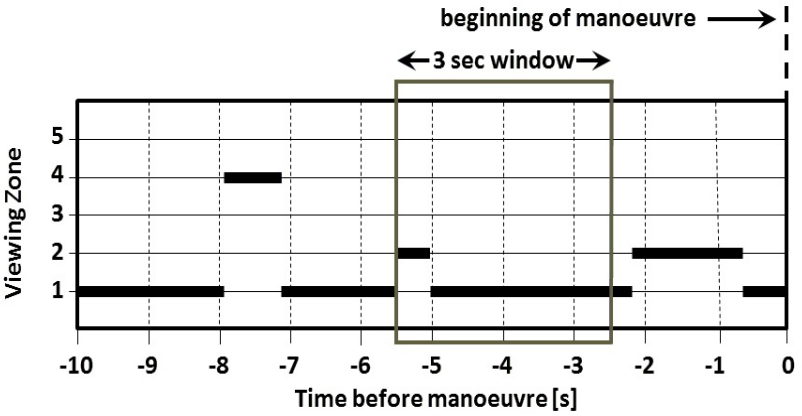


Fig. 3. Moving window applied to gaze data used from a section of typical gaze patterns prior to a LCL manoeuvre

The data in the selected windows was encoded into a format suitable for use in supervised learning, i.e. a vector of input and target data. The input section of the vector described the proportion of time spent looking in the five viewing zones within the selected window of data (i.e. the inputs summed to 1.0), the target section of the vector indicated the 'class' represented by the input vector and used a binary encoding, i.e. if the data represented lane keeping the target output was 0 and if it represented one of the lane change manoeuvres then the target was denoted as 1. The result was three groups of data each consisting of 284 (input, target) vectors of instances of LCL, LCR, and lane keeping.

4 Artificial Neural Networks

Feedforward ANNs with two hidden nodes and a single output node (which used the Sigmoid transfer function) were trained using the Backpropagation algorithm [8] to be binary classifiers ideally outputting 1 when an instance of the manoeuvre of interest was detected in the input data and 0 when the inputs indicated that the car was keeping to the lane. The weights were randomly initialised in the range $[-1.0, 1.0]$. The actual output of the ANNs was a real number in the range $[0.0, 1.0]$, which can be taken as a measure of probability that the manoeuvre of interest has been detected. A threshold $\mathcal{T}$, was then used in order to decide which of the two classes (lane change, lane keeping) the ANN output represented, i.e. the output of the ANN is $P(C|\mathbf{x})$ where C is the class lane change and $\mathbf{x}$ is the input vector, hence the threshold $\mathcal{T}$ can be used to decide which class the ANN output represents as follows:

IF $P(C|x) > \mathcal{T} \rightarrow$ lane change
ELSE $\rightarrow$ lane keeping

5 Analysis and Results

The aim of this study was not to establish where the ANN models obtained produced the best results. The aim was to establish the limits at which useful predictive models can be obtained, i.e. how far in advance of manoeuvre onset can predictive models produce results that are better than chance? What influence does the amount of data used have on the ability to predict? Here, the aim was to ascertain at what point the models produced cease to predict, i.e where the output reaches the limit of what would be obtained with random guessing. Balanced datasets were used, i.e. 50% lane change manoeuvre and 50% lane keeping manoeuvre, which meant that when the level of mean test classification error reached 0.5, the output was no better than random guessing. Fig.s 4 and 5 contain plots showing the effect of TBM and threshold value on the mean test error rate obtained for the selection of the data window sizes trialled. Table 1 summarises, for each data window size the maximum TBM and threshold value $\mathcal{T}$ at which the ANN model still produces predictions better than random guessing. By comparing the diagrams in Fig.s 4 and 5, which show the results for LCL and LCR respectively, a similar pattern of degradation in test error rate, i.e. it increases, can be seen to occur with changing $\mathcal{T}$ and increasing TBM. It should, however, be noted that the test error rate for LCR is greater for any given TBM and window size combination compared with the test error rate for LCL. Common to both LCL and LCR for any given TBM, is that the use of a larger window of data results in a lower test error rate. In addition, for any given window size, the values of $\mathcal{T}$ at which the lower test error rates are found, occur within a gradually smaller range (which centres at 0.5) as TBM increases.

Table 1. Maximum time before manoeuvre (TBM) and threshold value $\mathcal{T}$

Lane Change Left			Lane Change Right	
Window Size	TBM	$\mathcal{T}$	TBM	$\mathcal{T}$
0.5	3.5	0.5	2.5	0.5
1.0	4.0	0.5	2.5	0.5
1.5	5.0	0.5	3.0	0.5
2.0	5.0	0.5	3.0	0.5
2.5	5.0	0.5	3.0	0.5
3.0	5.0	0.5	3.0	0.5
3.5	5.0	0.5	3.0	0.5
4.0	5.0	0.5	3.0	0.5
4.5	5.0	0.45	3.0	0.5
5.0	5.0	0.45	3.0	0.5

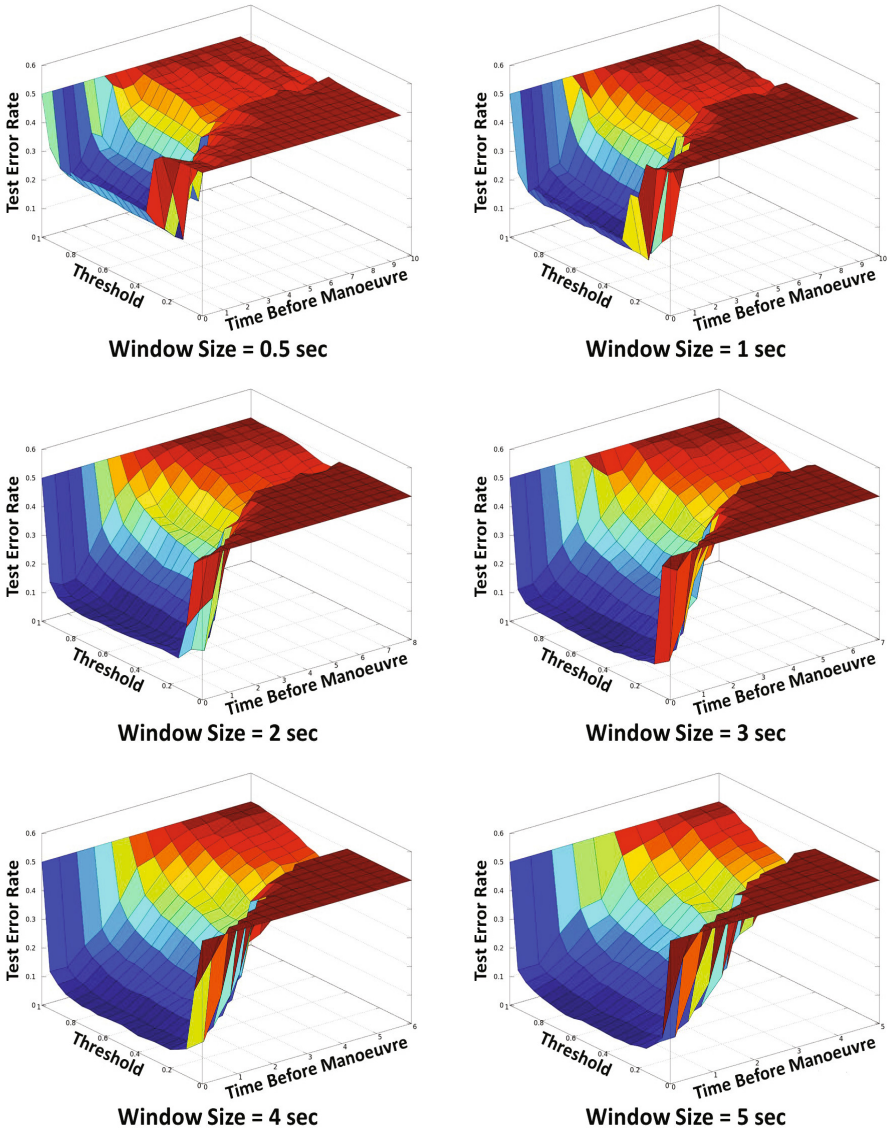


Fig. 4. Results for distinguishing Lane Change Left manoeuvres from Lane Keeping

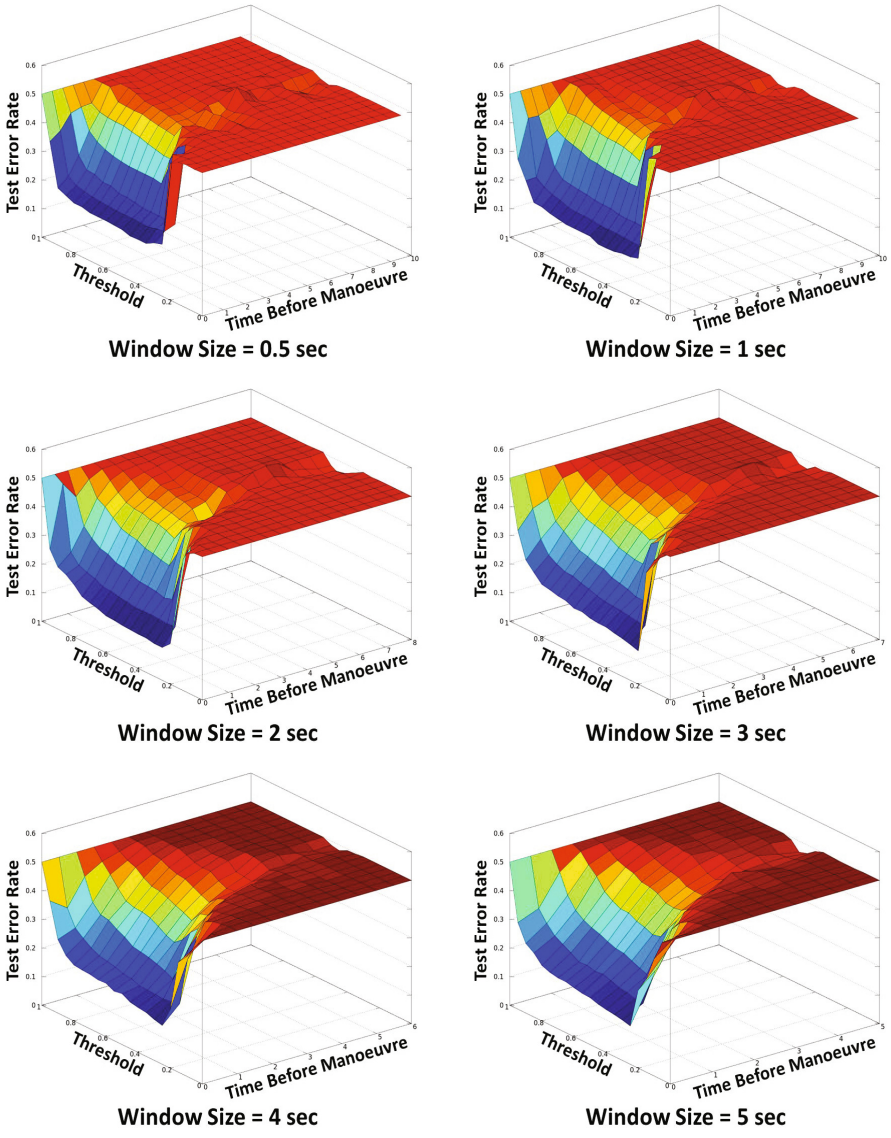


Fig. 5. Results for distinguishing Lane Change Right manoeuvres from Lane Keeping

6 Conclusion

This investigation has demonstrated again that gaze data is viable as a 'stand alone' data source for building predictive models of lane change behaviour. The ability of the ANNs to accurately classify the impending manoeuvre has been demonstrated to decline with increasing distance from the manoeuvre onset. It is, however, also noted that a level of predictive accuracy is maintained that is above that obtained by random guessing to a distance of 3.5 to 5 seconds before manoeuvre onset for distinguishing LCL from Lane Keeping and up to a distance of 2.5 to 3 seconds when distinguishing LCR from Lane Keeping. It was noted in the previous study that LCR is harder to predict and it is theorised that this is due to the fact that the distinctive gaze behaviour occurs closer to the time at which manoeuvre onset occurs than in the case of LCL. Increasing the size of the window of data has been shown to decrease the level of error for all times before manoeuvre onset. This is likely due to the fact that when a larger window of data is used the chances of capturing the pertinent gaze data are increased allowing accurate predictions to be made. Future work will also seek to combine gaze data with other available car data and investigate other methods of encoding the input data in order to produce models with improved predictive capability at the largest TBM onset that is possible.

References

1. Lethaus, F., Rataj, J.: Do eye movements reflect driving manoeuvres? IET Intelligent Transport Systems 1(3), 199–204 (2007)
2. Lethaus, F., Rataj, J.: Using eye movements as a reference to identify driving manoeuvres. In: Proceedings of the FISITA World Automotive Congress 2008, vol. 1, Springer Automotive Media, Wiesbaden (2008)
3. Lethaus, F.: Using eye movements to predict driving manoeuvres. In: Europe Chapter of the Human Factors and Ergonomics Society Annual Conference, Linköping, Sweden (2009)
4. Lethaus, F., Baumann, M.R.K., Köster, F., Lemmer, K.: Using Pattern Recognition to Predict Driver Intent. In: Dobnikar, A., Lotrič, U., Šter, B. (eds.) ICANNGA 2011, Part I. LNCS, vol. 6593, pp. 140–149. Springer, Heidelberg (2011)
5. Henderson, J.M., Ferreira, F.: Scene perception for psycholinguists. In: Henderson, J.M., Ferreira, F. (eds.) The Interface of Language, Vision, and Action: Eye Movements and the Visual World, pp. 1–58. Psychology Press, New York (2004)
6. McCall, J.C., Wipf, D.P., Trivedi, M.M., Rao, B.D.: Lane change intent analysis using robust operators and sparse Bayesian learning. IEEE Tranactions on Intelligent Transportation Systems 8(3), 431–440 (2007)
7. Oliver, N., Pentland, A.P.: Graphical models for driver behavior recognition in a SmartCar. In: Proceedings of IEEE Conference on Intelligent Vehicles, Detroit, MI, USA (2000)
8. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning internal representations by error propagation. In: Rumelhart, D.E., McClelland, J.L. (eds.) Parallel Distributed Processing, vol. 1, pp. 318–362. MIT Press, Cambridge (1986)

Particle Swarm Optimization for Auto-localization of Nodes in Wireless Sensor Networks

Stefania Monica and Gianluigi Ferrari

Wireless Ad-hoc and Sensor Networks Laboratory
Department of Information Engineering
University of Parma, 43124 Parma, Italy
stefania.monica@studenti.unipr.it, gianluigi.ferrari@unipr.it
http://wasnlab.tlc.unipr.it/

Abstract. In this paper, we consider the problem of auto-localization of the nodes of a static Wireless Sensor Network (WSN) where nodes communicate through Ultra Wide Band (UWB) signaling. In particular, we investigate auto-localization of the nodes assuming to know the position of a few initial nodes, denoted as "beacons". In the considered scenario, we compare the location accuracy obtained with the widely used Two-Stage Maximum-Likelihood algorithm with that achieved with an algorithm based on Particle Swarming Optimization (PSO). Accurate simulation results show that the latter can significantly outperform the former.

Keywords: Auto-localization, Particle Swarm Optimization, Maximum-Likelihood Algorithms.

1 Introduction

The location of sources in an indoor environment is of great interest because it has applications in many areas, such as monitoring of people in hospitals or in high security areas, search of victims or firefighters in emergency situations, home security, and finding people or vehicles in a warehouse. Wireless Sensor Networks (WSNs) are a leading option to address this problem, since they combine low to medium rate communications with positioning capabilities [1]. As a matter of fact, radio signal exchanges between nodes enables them to estimate the distance to each other, by extracting some physical quantities, such as the Received Signal Strength (RSS), the Angle Of Arrival (AOA), or the Time Of Flight (TOF) from the signals travelling between them.

Assuming to know the exact positions of a sufficiently large number of nodes, the position of a new node can be estimated by measuring its distances from a few nodes with known positions. Of course, wireless communications are affected by noise, especially in indoor environments, because of non-line-of-sight, multipath and multiple access interference. Ultra Wide Band (UWB) signaling

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 456–465, 2013.

seems a promising technology in this area, since the large bandwidth allows to resolve multipath components and the high time resolution improves their ranging capability (thus making the position estimate more accurate) [2].

In this paper, we consider an indoor scenario, which may be, for instance, a warehouse, in which a certain number of fixed Anchor Nodes (ANs) is used to locate Target Nodes (TNs), such as moving people and vehicles. In order to guarantee accurate TN estimation in every accessible point inside the building, which might be very large, a huge number of accurately positioned ANs would be necessary and this could be very demanding also from an economic point of view. Moreover, if the geometry of the warehouse changes, (e.g., for varying quantities of stored goods) the ANs might be replaced and/or new fixed ANs might be added. In order to overcome this problem, we consider auto-localization of the ANs under the assumption of initially knowing exactly the positions of only a few ANs, denoted as "beacons". UWB signaling is used and we consider a TOF approach to estimate the distances between pairs of nodes. In particular, we focus on a Time Difference Of Arrival (TDOA) approach, which is based on the estimation of the difference between the arrival times of signals traveling between each node to locate and nodes with known position (beacons or nodes whose position has already been estimated).

Many location estimate techniques, based on range measurement, have been proposed in the literature. Among them, it is worth recalling iterative methods, such as those based on Taylor-series expansion [3], or the steepest-descent algorithm [4]. These methods guarantee fast convergence for an initial value close to the true solution (which is often difficult to obtain in practice), but they are computationally expensive and convergence is not guaranteed, since, for instance, ignoring higher order terms in the Taylor-series expansion may lead to significant errors. To overcome these limitations, closed-form algorithms have been studied, such as least-square methods, approximated maximum-likelihood method [5], the plane intersection method [6] and the Two-Stages Maximum-Likelihood (TSML) method [7] [8]. In particular, the TSML method is one of the most commonly used, since it has been proved that it can attain the Cramer-Rao lower bound [9]. By observing that the initial system of equations of the TSML can be re-interpreted as an optimization problem, we thus solve it through the use of Particle Swarming Optimization (PSO). The proposed approach is shown to perform better than the TSML method.

This paper is organized as follows. In Section 2 the TSML method and the PSO algorithm are described. In Section 3 numerical results are presented. Section 4 concludes the paper.

2 Description of the Scenario

Throughout the paper, it is assumed that all the ANs lay on a plane, e.g., the ceiling of a warehouse. As anticipated in Section 1, a sufficient (but small) number of known "beacons" is used to get the position estimate of each AN with unknown position. Define: $\underline{s}_i = [x_i, y_i]^T, \forall i = 1, \ldots, M$ as the (known) positions

of M ANs (the "beacons"); $\underline{u}_e = [x_e, y_e]^T$ as the true position of a generic AN whose position needs to be estimated; $\hat{\underline{u}}_e = [\hat{x}_e, \hat{y}_e]^T$ as its estimated position. The true and estimated distances between the $i-$th beacon and the AN with position to be estimated are, respectively:

$$r_i = \sqrt{(\underline{u}_e - \underline{s}_i)^T(\underline{u}_e - \underline{s}_i)} \qquad \hat{r}_i = \sqrt{(\hat{\underline{u}}_e - \underline{s}_i)^T(\hat{\underline{u}}_e - \underline{s}_i)}. \tag{1}$$

As we are considering UWB signaling, it can be shown that $\hat{r}_i = r_i + \nu_i$, where $\nu_i = \varepsilon_i + b$ where $\varepsilon_i \sim \mathcal{N}(0, \sigma_i^2)$ and ε_i is independent from ε_j if $i \neq j$, and b is the synchronization bias [10]. Moreover, according to [10], the standard deviation of the position error estimation between two UWB nodes can be approximated as a linear function of the distance between them, namely

$$\sigma_i = \sigma_0 r_i + \beta. \tag{2}$$

In the following, the values $\sigma_0 = 0.01$ m and $\beta = 0.08$ m are considered. These values are obtained in [10] by considering Channel Model 3 described in [11] and the energy detection receiver presented in [12], which is composed by a band-pass filter followed by a square-law device and an integrator, in which the integration interval was set equal to $T_s = 1$ s. Therefore, the results presented in the following hold under these channel and receiver conditions.

2.1 TSML Method

The position estimation is carried on by using a simplification of the two step algorithm proposed in [8]. Note that at least 4 ANs positions must be known in order to start applying the following algorithm. Defining $\Delta_{1i} = r_i - r_1$, $\forall\ i = 2, \ldots, M$, and observing that $r_i^2 = (\Delta_{1i} + r_1)^2$, the following TDOA non-linear equations can be derived:

$$\Delta_{1i}^2 + 2\Delta_{1i} r_1 = -2x_i x_e - 2y_i y_e + x_e^2 + y_e^2 - K_i - r_1^2 \qquad i = 2, \ldots, M \tag{3}$$

where $K_i = x_i^2 + y_i^2$. When using estimated distances instead of real ones, the set of equations (3) can be written as

$$\underline{\underline{\hat{G}}}\, \hat{\underline{u}}_e = \hat{\underline{h}} \tag{4}$$

where

$$\underline{\underline{\hat{G}}} = -\begin{pmatrix} x_2 - x_1 & y_2 - y_1 & \hat{\Delta}_{12} \\ x_3 - x_1 & y_3 - y_1 & \hat{\Delta}_{13} \\ \vdots & \vdots & \vdots \\ x_M - x_1 & y_M - y_1 & \hat{\Delta}_{1M} \end{pmatrix} \qquad \hat{\underline{h}} = \frac{1}{2}\begin{pmatrix} K_1 - K_2 + \hat{\Delta}_{12}^2 \\ \vdots \\ K_1 - K_M + \hat{\Delta}_{1M}^2 \end{pmatrix} \tag{5}$$

and $\hat{\Delta}_{1i} = \hat{r}_i - \hat{r}_1$, $\forall i = 2, \ldots, M$. The system (4) would be a linear system in the three unknowns x_e, y_e and r_1 if r_1 did not depend on x_e and y_e (which is

instead the case, since by definition $r_1 = \sqrt{(x_e - x_1)^2 + (y_e - y_1)^2}$). In order to take into account this dependence, the solution is determined in 2 steps. First, it is supposed that x_e, y_e, and r_1 are three independent variables and the (linear) system is solved by using the Least Square (LS) technique. Defining $\underline{\hat{\phi}}_1 = [\hat{x}_e, \hat{y}_e, \hat{r}_1]^T$ and $\underline{\phi}_1 = [x_e, y_e, r_1]^T$ leads to the error vector

$$\underline{\psi} = \hat{\underline{\underline{G}}}(\underline{\hat{\phi}}_1 - \underline{\phi}_1). \tag{6}$$

The Maximum Likelihood (ML) solution of (6) is

$$\underline{\hat{\phi}}_1 = (\hat{\underline{\underline{G}}}^T \underline{\underline{\Psi}}^{-1} \hat{\underline{\underline{G}}})^{-1} \hat{\underline{\underline{G}}}^T \underline{\underline{\Psi}}^{-1} \underline{\hat{h}} \tag{7}$$

where

$$\underline{\underline{\Psi}} = \mathbb{E}[\underline{\psi}\,\underline{\psi}^T] = \underline{\underline{B}}\,\underline{\underline{Q}}\,\underline{\underline{B}}, \tag{8}$$

and $\underline{\underline{B}} = \mathrm{diag}(r_2, \ldots, r_M)$, $\underline{\underline{Q}} = \mathbb{E}[\underline{\varepsilon}_1 \underline{\varepsilon}_1^T]$ where $(\underline{\varepsilon}_1)_j = \hat{\Delta}_{1j} - \Delta_{1j}$ [8]. Omitting non linear perturbation, from (7) one obtains

$$\underline{\hat{u}}_e - \underline{u}_e = (\underline{\underline{G}}^T \underline{\underline{\Psi}}^{-1} \underline{\underline{G}})^{-1} \underline{\underline{G}}^T \underline{\underline{\Psi}}^{-1} (\underline{\hat{h}} - \hat{\underline{\underline{G}}}\,\underline{u}_e) = (\underline{\underline{G}}^T \underline{\underline{\Psi}}^{-1} \underline{\underline{G}})^{-1} \underline{\underline{G}}^T \underline{\underline{\Psi}}^{-1} (\underline{\psi}).$$

Since $\mathbb{E}[\underline{\psi}] = \underline{0}$, it can be noticed that $\mathbb{E}[\underline{\hat{u}}_e] = \underline{u}_e$ and the covariance matrix of $\underline{\hat{\phi}}_1$ is then [9]

$$\mathrm{cov}(\underline{\hat{\phi}}_1) \triangleq \mathbb{E}[(\underline{\hat{\phi}}_1 - \underline{\phi}_1)(\underline{\hat{\phi}}_1 - \underline{\phi}_1)^{\mathrm{T}}] = (\underline{\underline{\mathrm{G}}}^{\mathrm{T}} \underline{\underline{\Psi}}^{-1} \underline{\underline{\mathrm{G}}})^{-1}. \tag{9}$$

Now taking into account the relation between x_e, y_e, and r_1 the following set of equations is obtained:

$$\underline{\psi}' = \underline{h}' - \underline{\underline{G}}' \underline{\phi}_2 \tag{10}$$

where

$$\underline{h}' = [([\underline{\hat{\phi}}_1]_1 - x_1)^2, ([\underline{\hat{\phi}}_1]_2 - y_1)^2, [\underline{\hat{\phi}}_1]_3^2]^T \qquad \underline{\underline{G}}' = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}$$

$$\underline{\phi}_2 = [(x_e - x_1)^2, (y_e - y_1)^2]^T.$$

The ML solution of (10) is

$$\underline{\hat{\phi}}_2 = (\underline{\underline{G}}'^T \underline{\underline{\Psi}}'^{-1} \underline{\underline{G}}')^{-1} \underline{\underline{G}}'^T \underline{\underline{\Psi}}'^{-1} \underline{h}' \tag{11}$$

where $\underline{\underline{\Psi}}' \triangleq \mathbb{E}[\underline{\psi}' \underline{\psi}'^T] = 4\underline{\underline{B}}' \mathrm{cov}(\underline{\hat{\phi}}_1) \underline{\mathrm{B}}'$, $\underline{\underline{B}}' = \mathrm{diag}(x_e - x_1, y_e - y_1, r_1)$ and [9]

$$\mathrm{cov}(\underline{\hat{\phi}}_2) = (\underline{\underline{\mathrm{G}}'}^{\mathrm{T}} \underline{\underline{\Psi}}'^{-1} \underline{\underline{\mathrm{G}}'})^{-1}. \tag{12}$$

This leads to

$$\underline{u}_e = \underline{\underline{U}} \left[\sqrt{[\underline{\hat{\phi}}_2]_1}, \sqrt{[\underline{\hat{\phi}}_2]_2} \right]^T + \underline{s}_1$$

where $\underline{\underline{U}} = \text{diag}[\text{sgn}(\hat{\underline{\phi}}_1 - \underline{s}_1)]$. The covariance matrix of the error is then given by

$$\underline{\underline{\Phi}} = \frac{1}{4}(\underline{\underline{B}}''^{-1}\text{cov}(\hat{\underline{\phi}}_2)\underline{\underline{B}}''^{-1}) \tag{13}$$

where $B'' = \text{diag}(\underline{u} - \underline{s}_1)$ [9].

2.2 PSO Algorithm

The starting point for the previous method was the system (4) defined in Section 2.1. Through simple algebraic manipulations, it can be written as

$$\underline{\underline{A}}\,\hat{\underline{u}}_e = \hat{\underline{t}} \tag{14}$$

where

$$\underline{\underline{A}} = -2\begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_M - x_1 & y_M - y_1 \end{pmatrix} \qquad \hat{\underline{t}} = \begin{pmatrix} \hat{r}_2^2 - \hat{r}_1^2 + K_1 - K_2 \\ \hat{r}_3^2 - \hat{r}_1^2 + K_1 - K_3 \\ \vdots \\ \hat{r}_M^2 - \hat{r}_1^2 + K_1 - K_M \end{pmatrix}. \tag{15}$$

Notice that in this way, the measurements affected by noise only appear in vector $\hat{\underline{t}}$, while matrix $\underline{\underline{A}}$ cointains known parameters. On the contrary, in (4) both the matrix $\hat{\underline{\underline{G}}}$ and the vector $\hat{\underline{h}}$ contain noisy data. The solution of the system (14) can be found by formulating it as an optimization problem, with the following solution:

$$\underline{u}_e = \text{argmin}_{\underline{u}} F(\underline{u}) \tag{16}$$

where

$$F(\underline{u}) \triangleq ||\hat{t} - \underline{\underline{A}}\,\underline{u}||.$$

To solve this problem, the PSO algorithm, introduced in [13], can be used. According to this algorithm, the set of potential solutions of an optimization problem can be modeled as a swarm of particles, and the aim is to produce computational intelligence (thus to guide all the particles towards the optimal solution of the given problem), by exploiting social interactions between individuals [14]. It is assumed that the swarm is composed by S individuals, and that every particle i, $i = 1, \ldots, S$ at any given instant t is associated with a position $\underline{x}^i(t)$ in the region of interest and with a velocity $\underline{v}^i(t)$, which are both randomly initialized at the beginning with values $\underline{x}^i(0)$ and $\underline{v}^i(0)$ and which are updated at each iteration [15]. Moreover, it is supposed that the entire system has a memory, which allows each particle to know, at every step, not only its own best position reached so far, but also the best position among the ones reached by any other particle in the swarm (or by any other particle in a given neighbourhood of the swarm) in previous iterations. Each particle also keeps track of the values of the function to optimize corresponding both to its best position and to the global

best position. These values are used to update the velocity (and thus the position) of every particle in each step. The updating rule for the velocity of particle i is [16]

$$\underline{v}^i(t+1) = \omega(t)\underline{v}^i(t) + c_1 R_1(t)(\underline{y}^i(t) - \underline{x}^i(t)) + c_2 R_2(t)(\underline{y}(t) - \underline{x}^i(t)) \qquad i = 1, \ldots, S \tag{17}$$

where $\omega(t)$ is a weight called *inertial factor*, c_1 and c_2 are positive real parameters called *cognition parameter* and *social parameter*, respectively, $R_1(t)$ and $R_2(t)$ are random variable drawn at each step from the uniform $(0, 1)$ distribution and $\underline{y}^i(t)$ and $\underline{y}(t)$ are the position of the $i-$th particle with the best objective function and the position of the best (among all particles) objective function reached until instant t [14]. They can be described as

$$\begin{aligned} \underline{y}^i(t) &= \text{argmin}_{\underline{z} \in \{\underline{x}^i(0), \ldots \underline{x}^i(t)\}} F(\underline{z}) \\ \underline{y}(t) &= \text{argmin}_{\underline{z} \in \{\underline{y}^1(t), \ldots \underline{y}^S(t)\}} F(\underline{z}). \end{aligned} \tag{18}$$

The meaning of formula (17) is to add to the previous velocity (which is weightened by means of a multiplicative factor) a stochastic combination of the direction to the best position of the $i-$th particle and to the best global position. The definition of the velocity given in (17) is then used to update the position of the $i-$th particle, according to the rule

$$\underline{x}^i(t+1) = \underline{x}^i(t) + \underline{v}^i(t) \qquad i = 1, \ldots, S.$$

This process is iterated until a stopping criterion (which might be the achievement of a satisfying value of F or a maximum number of iterations) is met. The position of the particle which best suits the optimization requirements in the last iteration is then considered as the optimal solution.

In Section 3, this algorithm is applied to the function $F(\underline{u})$ defined in (16). The number of particles in the simluations is $S = 40$, and the number of iterations has been set to 50. The parameters c_1 and c_2 in (17) are both set equal to 2, which is a recommended choice since it makes the weights for social and cognition parts to be 1 on average [13]. The inertial factor $\omega(t)$ has been chosen to be a decreasing function in the number of iterations. As a matter of fact, a large value of the inertial factor corresponds to low dependece of the solution on initial population, and any good optimization algorithm should possess more exploitation ability at the beginning. Decreasing the value of $\omega(t)$ reduces the capability of PSO to exploit new areas, thus making the method more simliar to a local search as the number of iteration increases, which is a good property for an optimization algorithm. In the following simulations it is assumed that the initial value of the inertial factor is $\omega(0) = 0.9$ and that it decreases linearly to 0.5, reached in the last iteration.

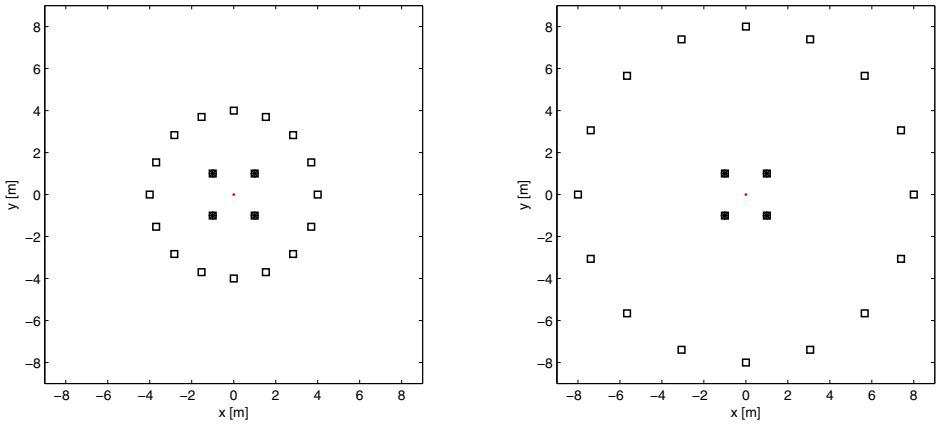
3 Simulation Results

In this section, we compare, through simulations, the two localization approaches, namely TSML and PSO, described in Section 2. The performances of the

algorithms are evaluated in terms of the Mean Square Error (MSE), which is defined as follows:

$$\text{MSE} \triangleq \mathbb{E}[(\hat{x}_e - x_e)^2 + (\hat{y}_e - y_e)^2]. \tag{19}$$

While in [17] the impact of the number of beacons is investigated, we now want to investigate the impact of the distance between beacons and ANs to be estimated. Two scenarios are considered, in which 4 beacons are used to estimate 16 unknown ANs around them. In the first scenario, the distance between the baricenter of the beacons and the remaining ANs is 4 m, while in the second case a longer distance (8 m) is considered. This two scenarios are shown in Fig. 1 (a.) and Fig. 1 (b.), respectively.



(*a.*) (*b.*)

Fig. 1. Possible scenarios where beacons (*full squares*) and ANs with unknown position (*empty squares*) are represented. The distance between the baricenter of the beacons (*red dot*) and the ANs is 4 m in (a.), and 8 m in (b.).

Fig. 2 (a.) represents the MSE relative to each AN when using only the beacons to get the location estimate. It can be noticed that the TSML method is far too unreliable, while the accuracy obtained using the PSO algorithm is satisfactory. The resulting estimated positions of the ANs are represented in Fig. 2 (b.), both in case of TSML method and PSO algorithm. Instead of considering only the beacons, a possible idea which can improve the accuracy is to consider the already estimated ANs as known ones, and to use them to calculate the position of the remaining nodes. Fig. 2 (c.) represents the MSE relative to each AN when using this second strategy and comparing the results with the ones of Fig. 2 (a.) shows that performances are improved. In particular, the accuracy of the location estimate when using TSML method significantly improves as the number of ANs assumed to be known increases. Less significant but still remarkable improvement can be noticed also in the behaviour of the MSE when using the PSO algorithm.

Notice that in both cases the PSO gives a better approximation of the real position of the ANs. The position estimates of the ANs obtained when following this strategy are represented in Fig. 2 (d.).

Fig. 3 represents the analogous results when the scenario is the one described in Fig. 1 (b.), so when the ANs are further from the beacons. Once again, from Fig. 3 (a.) and Fig. 3 (c.) it can be noticed that the PSO algorithm outperforms the TSML mehtod. Comparing these results with the analogous ones represented in Fig. 2 (a.) and Fig. 2 (c.), respectively, shows that a bigger distance between beacons and ANs leads to worst accuracy, especially when the TSML method is used. As a matter of fact, even when also the already estimated ANs are used to localize the remaining ones (as in Fig. 3 (c.)), the accuracy obtained with this method is not satisfactory. On the other hand, the accuracy obtained when using the PSO algorithm is still good, and in this case the distance between ANs does not seem to impact much on the solutions. Fig. 3 (b.) and Fig. 3 (d.) represent the obtained position estimate, analogous to Fig. 2 (b.) and Fig. 2 (d.).

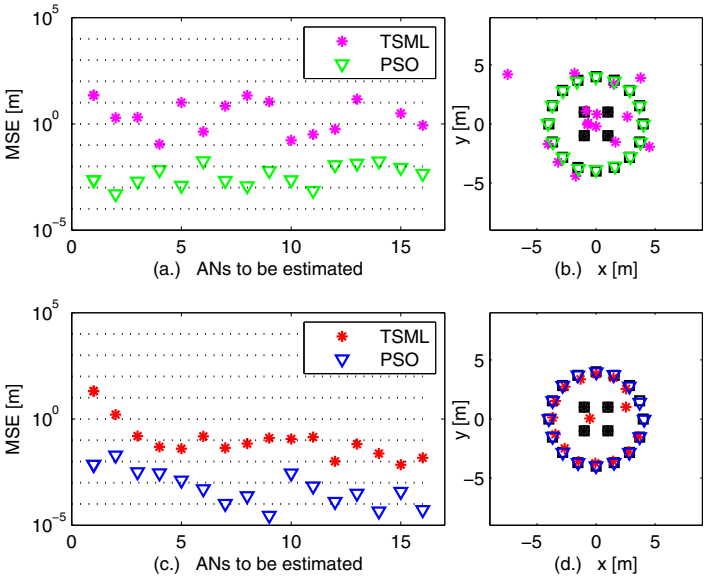


Fig. 2. The MSE of the ANs relative to the scenario described in Fig. 1 (a.) is plotted in (a.) and (c.). Fig. 2 (a.) refers to the case when only the beacons are used to estimate the position of all the ANs, both when TSML method is used (*magenta dots*) and when PSO algorithm is used (*green triangles*). The resulting position estimate are represented in Fig. 2 (b.), both in case of TSML method (*magenta dots*) and PSO algorithm (*green triangles*). Fig. 2 (c.) refers to the case in which also already estimated ANs are used to get the position of the remaining ones, both when TSML method is used (*red dots*) and when PSO algorithm is used (*blue triangles*). The resulting position estimate are represented in Fig. 2 (d.), both in case of TSML method (*red dots*) and PSO algorithm (*blue triangles*).

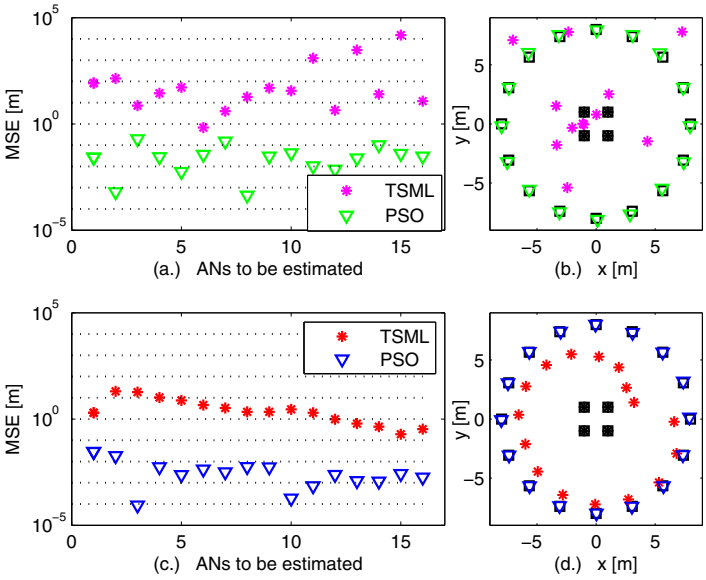


Fig. 3. The MSE of the ANs relative to the scenario described in Fig. 1 (a.) is plotted in (a.) and (c.). Fig. 2 (a.) refers to the case when only the beacons are used to estimate the position of all the ANs, both when TSML method is used (*magenta dots*) and when PSO algorithm is used (*green triangles*). The resulting position estimate are represented in Fig. 2 (b.), both in case of TSML method (*magenta dots*) and PSO algorithm (*green triangles*). Fig. 2 (c.) refers to the case in which also already estimated ANs are used to get the position of the remaining ones, both when TSML method is used (*red dots*) and when PSO algorithm is used (*blue triangles*). The resulting position estimate are represented in Fig. 2 (d.), both in case of TSML method (*red dots*) and PSO algorithm (*blue triangles*).

4 Conclusion

Two different approaches to UWB-signaling-based auto-localization of nodes in a static WSN have been considered. Besides solving the non-linear system of the localization equations by means of the TSML method, which is widely used for this kind of application, the trasformation of the original problem into an optimization one allows to solve it by means of the PSO algorithm. Our results show that the novel approach based on the use of the PSO algorithm allows to achieve better accuracy in the position estimate.

Acknowledgment. This work is supported by Elettric80 (http://www.elettric80.it).

References

1. Gezici, S., Poor, H.V.: Position estimation via ultra-wide- band signals. Proc. IEEE 97(2), 386–403 (2009)
2. Zhang, J., Orlik, P.V., Sahinoglu, Z., Molisch, A.F., Kinney, P.: UWB systems for wireless sensor networks. Proc. IEEE 97(2), 313–331 (2009)
3. Wade, H.F.: Position-location solutions by Taylor-series estimation. IEEE Trans. Aerosp. Electron. Syst. AES-12(2), 187–194 (1976)
4. Mensing, C., Plass, S.: Positioning algorithms for cellular networks using TDOA. In: Proceedings of the 2006 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2006, vol. 4 (May 2006)
5. Shen, G., Zetik, R., Thomä, R.S.: Performance comparison of TOA and TDOA based location estimation algorithms in LOS environment. In: Proceedings of the 5th Workshop on Positioning, Navigation and Communication, WPNC 2008 (2008)
6. Schmidt, R.O.: A new approach to geometry of range difference location. IEEE Trans. Aerosp. Electron. Syst. AES-8(6), 821–835 (1972)
7. Chan, Y., Ho, K.C.: A simple and efficient estimator for hyperbolic location. IEEE Trans. Signal Process 42(8), 1905–1915 (1994)
8. Ho, K.C., Xu, W.: An accurate algebraic solution for moving source location using TDOA and FDOA measurements. IEEE Trans. Signal Process. 52(9), 2453–2463 (2004)
9. Ho, K.C., Lu, X., Kovavisaruch, L.: Source localization using TDOA and FDOA measurements in the presence of receiver location errors: analysis and solution. IEEE Trans. Signal Process. 55(2), 684–696 (2007)
10. Busanelli, S., Ferrari, G.: Improved ultra wideband-based tracking of twin-receiver automated guided vehicles. Journal of Integrated Computer-Aided Engineering 19(1), 3–22 (2012)
11. Molisch, A.F., Cassioli, D., Chong, C.-C., Emami, S., Fort, A., Kannan, B., Karedal, J., Kunisch, J., Schantz, H.G., Siwiak, K., Win, M.Z.: A comprehensive standardized model for ultrawideband propagation channels. IEEE Trans. Antennas Propagat. 54(11), 3151–3166 (2006)
12. Dardari, D., Chong, C.C., Win, M.Z.: Threshold-based time-of-arrival estimators in uwb dense multipath channels. IEEE Trans. Commun. 56(8), 1366–1378 (2008)
13. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proc. IEEE International Conf. on Neural Networks, Perth, Australia, IEEE Service Center, Piscataway (1995)
14. Poli, R., Kennedy, J., Blackwell, T.: Particle swarm optimization. Swarm Intelligence Journal 1(1) (2007)
15. Eberhart, R., Kermedy, J.: A new optimizer using particles swarm theory. In: Proc. Sixth International Symposium on Micro Machine and Hmm Science, Nagoya, Japan, IEEE Service Center, Piscataway (1995)
16. Shi, Y., Eberhart, R.: A modied particle swarm optimizer. In: Proc. IEEE International Conference on Evolutionary Computation, Piscataway, NJ, pp. 69–73 (1999)
17. Monica, S., Ferrari, G.: Impact of the number of beacons in PSO-based autolocalization in UWB networks. To appear in Proceedings of EvoApplications (2013)

Effective Rule-Based Multi-label Classification with Learning Classifier Systems

Miltiadis Allamanis[1], Fani A. Tzima[2], and Pericles A. Mitkas[2,3]

[1] University of Edinburgh, Edinburgh, EH8 9AB, UK
[2] Aristotle University of Thessaloniki, 541 24, Thessaloniki, Greece
[3] Information Technologies Institute, Centre for Research & Technology - Hellas
m.allamanis@ed.ac.uk, fani@issel.ee.auth.gr, mitkas@auth.gr

Abstract. In recent years, multi-label classification has attracted a significant body of research, motivated by real-life applications such as text classification and medical diagnoses. However, rule-based methods, and especially Learning Classifier Systems (LCS), for tackling such problems have only been sparsely studied. This is the motivation behind our current work that introduces a generalized multi-label rule format and uses it as a guide for further adapting the general Michigan-style LCS framework. The resulting LCS algorithm is thoroughly evaluated and found competitive to other state-of-the-art multi-label classification methods.

Keywords: multi-label classification, learning classifier systems, genetics-based machine learning, classification, evolutionary computation.

1 Introduction

One of the most extensively studied knowledge extraction tasks is classification, wherein problems can be either multi-class (single-label) or multi-label. Single-label problems, where data samples are associated with a single class, have been thoroughly explored, with various Machine Learning algorithms [1–3]. On the other hand, literature on multi-label classification – an extension of the single-label case, where each sample is associated with one or more categories, named labels – is far less abundant. Multi-label classification tasks are, however, common in real-life and, lately, there have been important developments, including algorithms for efficiently tackling such problems [4] and a wide range of applications, ranging from text classification and medical diagnoses to protein function prediction and the processing of semantic scenes.

Regardless of the kind of problem (single- or multi-label) being tackled, rule-based solutions to classification problems are highly desirable, due to the inherent ability of rule-inducing algorithms to provide human readable representations. In this direction, Learning Classifier Systems (LCS) [5], a Genetics-based Machine Learning method combining evolutionary computing and reinforcement [6] or supervised learning [7–9], can be an effective alternative. Indeed, in recent years, LCS have been modified for data mining [10] and single-step classification problems, notably in UCS [8], while an approach to multi-label classification with LCS has been explored in [11] with promising results.

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 466–476, 2013.

In the current work, we effectively tackle multi-label classification using rule-based methods and, more specifically, LCS. In this direction, we employ a supervised framework, partly based on UCS, and extend it, to render it directly applicable to multi-label classification, without the need for any problem transformation. We avoid the extra complexity of default hierarchies and organizational learning added in [11] and provide a more extensive evaluation. Our main contributions, detailed after briefly presenting the relevant background (Section 2), are: (i) a *generalized multi-label rule format* (Section 3) that has several distinct advantages over those used in other multi-label classification methods; (ii) a *multi-label Learning Classifier System* (Section 4), named MLUCS, whose components allow for efficient and accurate multi-label classification through expressive multi-label rulesets; and (iii) an experimental evaluation (Section 5) of our proposed LCS approach, against other state-of-the-art algorithms on widely used datasets, that validates its potential.

2 Background

2.1 Multi-label Classification

Multi-label classification is a generalization of the multi-class case where samples are associated with a set of mutually non-exclusive categories or labels $Y \subseteq L$. Thus, a multi-label classification model approximates $f\colon X \to L^*$ where X is the feature space and L^* is the powerset of the set of all labels L (label space).

There are two main approaches to multi-label classification [4]. **Problem transformation** methods transform a multi-label classification problem into a set of single-label ones and, thus, involve trade-offs between training time and label correlation representation. Various transformations have been proposed, including the Binary Relevance (BR), Ranking by Pairwise Comparison, and Label Powerset (LP) transformations, the Classifier Chains method [12], and RAkEL [13]. **Algorithm transformation** methods, on the other hand, adapt learning algorithms to directly handle multi-label data. Such methods include MLkNN [14], Adaboost.MH [15] and multi-label decision trees [16], as well as the LCS proposed in our current work.

For evaluating multi-label classifiers, several traditional metrics are used, after being properly modified. For a dataset D – consisting of multi-label instances of the form (x_i, Y_i), where $i = 1 \ldots |D|$ and $Y_i \subseteq L$ ($Y_i \in L^*$) – and a prediction function $\hat{Y}_i = H(x_i)$, **accuracy** is defined as the mean, over all instances, ratio of the size of the intersection and union sets of actual and predicted labels:

$$\text{ACCURACY}(H, D) = {}^{1}/_{|D|} \sum_{i=1}^{|D|} {}^{|Y_i \cap \hat{Y}_i|}/_{|Y_i \cup \hat{Y}_i|} \tag{1}$$

while **exact match** (or subset accuracy) is a simpler, yet stricter, metric, calculated as the label-set-based accuracy:

$$\text{EXACT-MATCH}(H, D) = {}^{|C|}/_{|D|} \tag{2}$$

where C is the set of correctly classified instances for which $Y_i \equiv \hat{Y}_i$.

2.2 Learning Classifier Systems

Learning Classifier Systems are an evolutionary approach to supervised and reinforcement learning. Since multi-label classification is a supervised task, we tackle the corresponding problems using *supervised LCS* following the "Michigan approach". Such LCS maintain a cooperative population of condition-action rules, termed classifiers, and combine supervised learning with a Genetic Algorithm (GA). The GA works on classifier conditions trying to adequately decompose the target problem into a set of subproblems, while supervised learning evaluates classifiers in each of them. The most prominent example of this class of systems is UCS, which we have, thus, chosen as the basis for our current work.

UCS is an accuracy-based LCS [7] that inherits the primary features of the "reinforcement learner" XCS [6], but specializes them for supervised learning. UCS employs a population of rules that gradually evolve through the interplay of various cooperating components: (i) an *Update Component* responsible for updating rule-specific parameters, such as accuracy and fitness; (ii) a *Discovery Component* exploring the search space and producing new rules through a steady-state GA; and (iii) a *Performance Component* exploiting the developed rules to classify previously unseen examples. UCS also employs fitness sharing [8]. The fitness F_{cl} of a classifier cl is computed by Eq. 3, where β, α, ν and acc_0 are user-defined parameters, $k^{'}_{cl}$ is the relative accuracy of cl, num_{cl} its numerosity and $\mathbf{M}$ a set of classifiers:

$$F_{cl} \leftarrow F_{cl} + \beta(k^{'}_{cl} - F_{cl}) \tag{3}$$

$$k^{'}_{cl} = \frac{k_{cl} \cdot num_{cl}}{\sum_{cl_i \in \mathbf{M}} k_{cl_i} \cdot num_{cl_i}} \text{ with } k_{cl \in \mathbf{C}} = \begin{cases} 1 & \text{if } acc_{cl} > acc_0 \\ a \cdot (acc_{cl}/acc_0)^{\nu} & \text{otherwise} \end{cases}$$

3 Rules for Multi-label Classification

To tackle multi-label classification with rule-based methods, such as LCS, we need an expressive rule format, able to capture correlations both between the features and labels and among the labels. In this Section, we introduce a format with such properties that forms the basis of MLUCS detailed in Section 4.

3.1 Generalized Multi-label Rule Representation

Single-label classification rules traditionally follow the production system form r_i: $condition_i \rightarrow Y_i$, where $condition_i$ comprises a conjunction of tests on attribute values and the consequent Y_i contains a single label $Y_i = \{y_i \mid y_i \in L\}$. For zero-order rules the condition takes the form $(X_1 \text{op}_1 u_1) \wedge \cdots \wedge (X_k \text{op}_k u_k)$, where X_i is an attribute, op_i an operator, u_i a constant set, number or range of numbers, and $0 \leq k \leq |X|$.

For rules to handle multi-label classifications, we modify the consequent Y_i, such that Thus, each rule may advocate for (l_i=1) or be opposed to (l_i=0)

a certain label, but it may also be agnostic (not know/care) about it (l_i=#). In other words, the modified rule format is capable of mapping conditions to arbitrary label subspaces.

3.2 Generalized Multi-label Rule(set) Properties

Rules following our proposed *Generalized Multi-label Representation* have two unique properties. They (i) are easy to interpret, rendering the discovered knowledge equally usable for humans and computers, and (ii) have a flexible label-correlation representation, i.e., they variably correlate the maximum possible number of labels to any given condition.

A ruleset R for single-label classification comprises *cooperative* rules that collectively solve the target problem, while being maximally *compact*, i.e., contain the minimum number of rules for solving the problem. Equivalently, all rules $r_i \in R$ have *maximally general conditions* and, thus, the greatest possible feature space coverage. Finally, a ruleset R is an effective solution if it contains rules that are adequately *correct*, given a specific performance/correctness metric. While all aforementioned properties remain desirable, *generalized multi-label rulesets additionally need to exhaustively cover the label space*. In other words, rules in a multi-label ruleset R should be able to collectively decide about all labels of every instance. This latter desirable property, together with the compactness requirement, indicates that multi-label rules should have maximally *general* conditions and maximally *specific* consequents. Thus, algorithms building multi-label rulesets have to consider the trade-off between condition generalization, consequent specialization and rule correctness.

4 LCS for Multi-label Classification

Our current work targets offline multi-label classification problems – i.e., problems that do not involve online interactions – which we tackle using a *"Michigan-style" supervised LCS*, named MLUCS. Based on Section 3.2, in this Section, we present and justify our design choices towards MLUCS.

4.1 Description of the MlUCS Algorithm

MLUCS employs a population **P** of gradually evolving, cooperative classifiers (rules) that collectively solve the target classification task, by each encoding a fraction of the problem. Associated with each classifier is a number of parameters: the *numerosity num*, i.e., the number of the classifier's copies (or microclassifiers) currently present in the ruleset; the *correct set size cs* that estimates the average size of the correct sets the classifier has participated in; the *time step ts* of the last occurrence of a GA in a correct set the classifier has belonged to; the *experience exp*, measured as the classifier's number of appearances in match sets; the *number of the classifier's correct (incorrect) decisions tp (fp)*; the *accuracy*

Algorithm 1. Training Cycle for MLUCS

```
1: MlUCSUpdate(P, Instance_j)
2: M ← generateMatchSet(P, l, Instance_j)
3: for each l ∈ L do
4:    C_l ← generateLabelCorrectSet(M, l, Instance_j)
5: for each rule ∈ M do
6:    updateFitness(rule)
7:    if ∃l_i ∈ L : rule ∈ C_{l_i} then updateCs(rule)
8:    rule.exp ← rule.exp + 1
9: for each l ∈ L do
10:   if C_l ≠ ∅ then evolve(C_l) else cover(Instance_j, l)
11: deleteIfNecessary(P)
```

acc that estimates the probability of a classifier predicting the correct class; and the *fitness* F that is a measure of the classifier's quality.

At each learning time-step t, MLUCS receives an instance X_i along with its labels Y_i ($X_i \rightarrow Y_i \mid Y_i \subseteq L$) and follows a cycle (summarized in Alg. 1) of *performance*, *update* and *discovery* component activation. Overall, the evolution of multi-label rules in MLUCS is iterative. First, a match set $\mathbf{M}$ is created containing the rules of the population $\mathbf{P}$ whose condition matches the current input. Next, for each label $l \in L$, a correct set $\mathbf{C}_l$ is formed containing the rules of $\mathbf{M}$ that correctly predict l for the current instance. Given these sets, rule parameters (including the fitness F) are updated per label and cs is updated only for those rules belonging to at least one C_l. Finally, the discovery component is activated (at most) once per label: for every non-empty $\mathbf{C}_l$ the `evolve()` function executes a steady-state GA that produces two offspring and adds them to the population $\mathbf{P}$. In case of an empty $\mathbf{C}_l$, the covering operator `cover()` creates a new rule based on the current input and appropriately generalizes it.

4.2 Components of the MlUCS Algorithm

Performance Component. Upon receipt of an instance $X_i \rightarrow Y_i$, MLUCS scans the population $\mathbf{P}$ for rules whose condition matches X_i and forms the *match set* $\mathbf{M}$. Next, for each label, a correct set $\mathbf{C}_l$ is formed containing the rules of $\mathbf{M}$ that correctly predict label l for the current instance. The rest of the classifiers in $\mathbf{M}$ are placed in the *incorrect set* $!\mathbf{C}_l$. If the system is in test mode[1], a classification decision is produced based on the labels advocated by rules in $\mathbf{M}$.

The classification of a new sample based on multi-label rules is not straightforward, because: (i) a bipartition of the label-set L, rather than a single class, has to be decided upon based on some threshold, while (ii) rulesets evolved with LCS may contain contradicting or low-accuracy rules. Therefore, a "vote and threshold" method is required [17]. More specifically, an overall vote w_l for each $l \in L$ is obtained by letting each rule vote according to its fitness. The votes vector

[1] Under test mode, the population does not undergo changes; that is, the update and discovery components are disabled.

Algorithm 2. Rule fitness and cs update for MLUCS

1: **updateFitness**($rule$)
2: **for each** $l \in L$ **do**
3: $rule.tp \leftarrow rule.tp + correctness(rule, l, Inst_j)$
4: $rule.msa \leftarrow rule.msa + msaValue(rule, l, Inst_j)$
5: $rule.acc \leftarrow {}^{rule.tp}/_{rule.msa}$
6: $k_{rule} = rule.num \cdot (rule.acc)^{\nu}$
7: $rule.F \leftarrow rule.F + \beta\,(k_{rule} - rule.F)$

1: **updateCs**(cl)
2: $minCs \leftarrow \underset{l \in L}{\operatorname{argmin}} \sum_{cl \in \mathbf{C}_l} cl.num$
3: $cl.cs \leftarrow cl.cs + \beta\,(minCs - cl.cs)$

w is normalized, such that $\sum_{l \in L} \bar{w}_l = 1$ and $w_l \in [0,1]$, $\forall l \in L$, and a threshold $t \in (0, 0.5]$ is selected. Finally, all labels l for which $\bar{w}_l \geq t$ are activated, yielding the required bipartition of labels. For MLUCS, we employed **internal validation** that searches for threshold values maximizing a metric (in our case the multi-label accuracy), based on consecutive internal tests.

Update Component. In training or explore mode, each classification of a data instance is associated with an update of the matching classifiers' parameters. More specifically: (i) for all classifiers belonging to at least one $\mathbf{C}_l$, their correct set size cs is updated, so that it estimates the average size of all correct sets the classifier has participated in so far; and (ii) for all classifiers in $\mathbf{M}$, their experience exp is increased by one and their fitness F is updated. The update strategies for fitness F and correct set size cs are implemented in the `updateFitness()` and `updateCs()` methods of Alg. 2.

Fitness calculation in MLUCS involves computing the accuracy (acc) of classifiers as the percentage of their correct classifications (lines 5-7 of Alg. 2). Moreover, motivated by the need to distinguish between rules that provide concrete decisions (positive or negative) and those that are "indifferent", we introduce the notion of *correctness* for rule decisions. The correctness value of a rule r for a label l and an instance i is calculated by the equation:

$$correctness(r, l, i) = \begin{cases} 1 \text{ if } r \text{ correctly predicts } l \text{ for } i \\ 0 \text{ if } r \text{ incorrectly predicts } l \text{ for } i \\ \omega \text{ if } r \text{ does not decide on } l \text{ for } i \end{cases} \quad \text{where } 0 \leq \omega \leq 1 \quad (4)$$

Equivalently, the match set appearances (msa) of a rule r for a label l and an instance i depends on whether r provides a concrete decision or not:

$$msaValue(r, l, i) = \begin{cases} \phi \text{ if } r \text{ does not decide on } l \text{ for } i \\ 1 \text{ otherwise} \end{cases} \quad \text{with } \phi \leq 1 \quad (5)$$

Regarding the update of the rule *overall correct-set size* (`updateCs()`), we use a strict estimation, employing the size of the smallest label correct set that the rule

participates in. The rationale of our choice is the need to exert fitness pressure in $\mathbf{P}$ towards complete label-space coverage, by rewarding rules that explicitly advocate for or against "unpopular" labels.

Discovery Component. MLUCS employs two rule discovery mechanisms: (i) a *steady-state niche genetic algorithm* and (ii) a *covering operator*.

The *genetic algorithm* (part of `evolve()` in Alg. 1) is applied iteratively on all correct sets $\mathbf{C}_l$ and invoked at a rate θ_{GA}, where θ_{GA} is a threshold on the average time since the last GA invocation of classifiers in $\mathbf{C}_l$. The evolutionary process employs parent selection based on tournaments of size $\tau_s{=}r{\cdot}|\mathbf{C}|$, $r{\in}(0,1)$. Two parent classifiers, selected based on their fitness, are copied to form two offspring after (single-point) crossover and mutation operators have been applied to them with probabilities χ and μ, respectively. Before insertion into the population $\mathbf{P}$, the offspring are checked for *subsumption* against each of their parents. If either of the parents is sufficiently experienced ($cl.exp{>}\theta_{sub}$), accurate ($cl.acc{>}acc_0$) and more general than the offspring, the latter is not introduced into $\mathbf{P}$, but the parent's numerosity is incremented by one instead. If the offspring are not subsumed by either parent, they are introduced into $\mathbf{P}$. The generality condition for subsumption is extended for the multi-label case, such that *a rule r_i can only subsume a rule r_j, if r_i's condition is equivalent or more general and its consequent is equivalent or more specific than those of the rule r_j being subsumed.*

The *covering operator* (`cover()` in Alg. 1) is activated only during training and introduces new rules to the population when the system encounters an empty $\mathbf{C}_l$. Covering produces one rule with a random condition matching the current input instance and generalized with a given probability $P_{\#}$ per locus. This process is followed by a generalization process applied to the consequent. All labels in the newly created rule's consequent are set to 0 or 1 according to the current input and then converted to "don't cares" with probability $P_{label\#}$ per label, except for the current label l that remains specific.

Overall, the discovery component is responsible for evolving the ruleset and creating new rules. The system, however, maintains a constant population size (at the microclassifier level) by employing a *deletion* mechanism. A rule cl is selected for deletion with probability P_{del} (δ and θ_{del} are user-defined):

$$cl.P_{del} = \frac{cl.d}{\sum_{cl_i \in \mathbf{P}} cl_i.d}, \text{ where } cl.d = \begin{cases} e^{-cl.cs} \cdot cl.F_{micro} & \text{if } cl.exp > \theta_{del} \\ \delta \cdot cl.F_{micro} & \text{otherwise} \end{cases} \quad (6)$$

5 Experimental Validation of the Proposed Approach

In this Section, we present an experimental evaluation of MLUCS against other state-of-the-art methods. We experiment with two versions of MLUCS: MLUCS_0 ($\phi{=}\omega{=}0$) does not penalize #s in the rule consequent, whereas $\text{MLUCS}_{\#}$ slightly penalizes "indifferent" rules by considering #s as partial ($\omega{=}0.9$) matches ($\phi{=}1$).

5.1 Experimental Setup

For all datasets[2] in Table 1, except *enron*, we performed ten-fold cross validation. For *enron*, because of its size, a single train-test evaluation was performed. The basic parameters used through all experiments for both MLUCS$_0$ and MLUCS$_\#$ were: μ=0.04, χ=0.8, β=0.2, ν=10, p=5, θ_{del}=20, θ_{sub}=10, acc_0=0.99 and $500 * |D|$ learning iterations. The population size $|P|$, number of iterations N, GA invocation rate θ_{GA} and generalization probabilities $P_\#$ and $P_{label\#}$ were varied per dataset, based on the latter's properties.

Table 1. Benchmark dataset statistics: number of instances $|D|$, number of nominal (C) or numeric (N) attributes $|X|$, number of labels $|L|$, number of distinct label combinations, label density and cardinality $|LCA|$.

Dataset	Domain	$\|D\|$	$\|X\|$		$\|L\|$	distinct	density	$\|LCA\|$
music	media	593	72	N	6	27	0.31	1.87
genbase	biology	662	1186	C	27	32	0.05	1.25
yeast	biology	2417	103	N	14	198	0.30	4.24
enron	text	1702	1001	C	53	753	0.06	3.38
CAL500	media	502	68	N	174	502	0.15	26.04

The rival algorithms against which the MLUCS algorithms are compared are BR-J48, RAkEL and MLkNN and, more specifically, their implementations provided by the *Mulan Library for Multi-label Learning*. BR-J48 is a binary-relation transformation using C4.5 and serves as our baseline. RAkEL [13] is a state-of-the-art method that transforms problems using k-sized label-sets, for which we used a subset size k=3, m=2L models and C4.5 as the base classifier. Finally, MLkNN [14] is a multi-label version of the well known "k-nearest-neighbors", for which we chose k=10 neighbors.

Finally, regarding the statistical significance of the measured differences in algorithm performance, we employ the procedure suggested by Demšar [18] for comparing classifiers across multiple datasets. This procedure involves a *Friedman test* to establish the significance of the differences between classifier ranks and, potentially, a post-hoc test to compare classifiers to each other. In our case, the evaluation goal is two-fold: i) compare the performance of all algorithms to each other and ii) compare the two versions of the proposed MLUCS algorithm. For the first goal the *Nemenyi test* was selected as the appropriate post-hoc test, while for the second we used the *Wilcoxon signed-rank test*.

5.2 Comparative Analysis of Results

Table 2 reports the results of the comparison of the studied MLUCS algorithms with their rivals, summarizing the obtained values for two evaluation metrics. Based on *accuracy*, the average rank provides a clear indication of the studied

[2] The datasets are available at the Mulan web-site: `http://mulan.sf.net/datasets.html`.

Table 2. Algorithm comparison. The best value per problem is marked in bold. Superscripts refer to algorithm ranks (per problem) according to the Friedman test. "Av" reports each algorithm's average rank and "Pos" its position in the final ranking.

(a) Algorithm evaluation based on the "Accuracy" metric

	music	genbase	yeast	enron	CAL500	Av.	Pos.
BR-J48	$46.23^{5.0}$	$98.62^{2.5}$	$43.95^{5.0}$	$36.71^{3.0}$	$20.67^{3.0}$	3.7	4
RAkEL-J48	$50.91^{4.0}$	$98.62^{2.5}$	$48.74^{4.0}$	$\mathbf{41.04}^{1.0}$	$\mathbf{22.94}^{1.0}$	2.5	2
MLkNN	$53.26^{3.0}$	$94.11^{5.0}$	$51.62^{2.0}$	$31.85^{5.0}$	$19.70^{5.0}$	4.0	5
MLUCS$_0$	$\mathbf{57.85}^{1.0}$	$\mathbf{98.87}^{1.0}$	$\mathbf{51.66}^{1.0}$	$38.13^{2.0}$	$21.24^{2.0}$	1.4	1
MLUCS$_\#$	$56.80^{2.0}$	$98.50^{4.0}$	$50.40^{3.0}$	$36.28^{4.0}$	$20.42^{4.0}$	3.4	3

(b) Algorithm evaluation based on the "Exact Match" metric

	music	genbase	yeast	enron	CAL500	Av.	Pos.
BR-J48	$18.38^{5.0}$	$97.13^{2.5}$	$6.83^{5.0}$	$8.64^{2.0}$	$0.00^{3.0}$	3.5	5
RAkEL-J48	$24.78^{4.0}$	$97.13^{2.5}$	$11.71^{4.0}$	$\mathbf{10.71}^{1.0}$	$0.00^{3.0}$	2.9	2
MLkNN	$28.31^{3.0}$	$90.64^{5.0}$	$\mathbf{18.74}^{1.0}$	$6.22^{3.0}$	$0.00^{3.0}$	3.0	3
MLUCS$_0$	$\mathbf{32.03}^{1.0}$	$\mathbf{97.58}^{1.0}$	$12.29^{3.0}$	$5.18^{4.0}$	$0.00^{3.0}$	2.4	1
MLUCS$_\#$	$30.38^{2.0}$	$96.37^{4.0}$	$14.15^{2.0}$	$2.59^{5.0}$	$0.00^{3.0}$	3.2	4

algorithms relative performance: MLUCS$_0$ ranks first, outperforming all its rivals in 3 out of the 5 studied problems, and MLUCS$_\#$ ranks third, performing better, on average, than the baseline BR-J48 method and MLkNN. As far as the two versions of our proposed MLUCS algorithm are concerned, obtained results indicate that their performance is similar, with MLUCS$_0$, however, outperforming MLUCS$_\#$ on all datasets. The comparison results are less favorable for MLUCS$_\#$ when based on *exact match*, as it ranks fourth. Still, MLUCS$_0$ remains the best performing algorithm, achieving the best exact match value for 2 out of the 4 problems with non-zero achievable metric values.

Regarding the statistical significance of the measured differences, the Friedman test does *not* reject the null hypothesis (at α=0.1) that all algorithms perform equivalently, when applied to rankings based on the exact match metric. On the other hand, the same null hypothesis is rejected (at α=0.05) when the studied algorithms are ranked based on accuracy, but the Nemenyi post-hoc test only detects a significant performance difference between MLUCS$_0$ and MLkNN. However, based on the more powerful Wilcoxon signed-ranks test, MLUCS$_0$ is found to perform better than 3 of its rivals, that is all except RAkEL, at a level of confidence greater than 90%. The same test confirms a statistically significant (α=0.1875) performance difference between MLUCS$_\#$ and MLkNN.

Overall, we consider the obtained results indicative of (i) the potential of MLUCS for effective multi-label classification, as well as (ii) the flexibility of the generalized multi-label rule format that can mimic the knowledge representations induced by both RAkEL and MLkNN, depending on the problem type.

6 Conclusions

In this paper we presented a *generalized rule format* for generating compact and accurate rulesets in multi-label settings, employing a *multi-label LCS algorithm* named MLUCS. MLUCS is based on a supervised learning framework, properly designed to meet the requirements posed by the multi-label classification domain. Its experimental evaluation revealed that it is capable of consistently effective classification, thus highlighting it as an interesting alternative to other multi-label classification methods. Major improvements in the training time of MLUCS are possible by exploiting the parallelization potential of genetic algorithms, while additional performance improvements may be achieved through the study of alternative approaches to fitness calculation.

References

1. Aha, D., Kibler, D., Albert, M.: Instance-based learning algorithms. Machine Learning 6(1), 37–66 (1991)
2. Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Proceedings of the Fifth Annual Workshop on Computational Learning Theory, pp. 144–152. ACM (1992)
3. Zhang, G.: Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C 30(4), 451–462 (2000)
4. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Mining multi-label data. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 667–685. Springer (2010)
5. Holland, J.: Adaptation. In: Rosen, R., Snell, F.M. (eds.) Progress in Theoretical Biology, vol. 4, pp. 263–293. Academic Press, New York (1976)
6. Wilson, S.W.: Classifier fitness based on accuracy. Evolutionary Computation 3(2), 149–175 (1995)
7. Bernadó-Mansilla, E., Garrell-Guiu, J.: Accuracy-based learning classifier systems: models, analysis and applications to classification tasks. Evolutionary Computation 11(3), 209–238 (2003)
8. Orriols-Puig, A., Bernadó-Mansilla, E.: Revisiting UCS: Description, Fitness Sharing, and Comparison with XCS. In: Bacardit, J., Bernadó-Mansilla, E., Butz, M.V., Kovacs, T., Llorà, X., Takadama, K. (eds.) IWLCS 2006 and IWLCS 2007. LNCS (LNAI), vol. 4998, pp. 96–116. Springer, Heidelberg (2008)
9. Tzima, F.A., Mitkas, P.A.: Strength-based learning classifier systems revisited: Effective rule evolution in supervised classification tasks. Engineering Applications of Artificial Intelligence 26(2), 818–832 (2013)
10. Bull, L., Bernadó-Mansilla, E., Holmes, J.H. (eds.): Learning Classifier Systems in Data Mining. SCI, vol. 125. Springer, Heidelberg (2008)
11. Vallim, R., Duque, T., Goldberg, D., Carvalho, A.: The multi-label OCS with a genetic algorithm for rule discovery: implementation and first results. In: Proceedings of GECCO 2009, pp. 1323–1330. ACM, New York (2009)
12. Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier Chains for Multi-label Classification. In: Buntine, W., Grobelnik, M., Mladenić, D., Shawe-Taylor, J. (eds.) ECML PKDD 2009, Part II. LNCS, vol. 5782, pp. 254–269. Springer, Heidelberg (2009)

13. Tsoumakas, G., Katakis, I., Vlahavas, I.P.: Random k-labelsets for multilabel classification. IEEE Transactions on Knowledge and Data Engineering 23(7), 1079–1089 (2011)
14. Zhang, M., Zhou, Z.: ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition 40(7), 2038–2048 (2007)
15. Schapire, R., Singer, Y.: Boostexter: A boosting- based system for text categorization. Machine learning 39(2), 135–168 (2000)
16. Vens, C., Struyf, J., Schietgat, L., Džeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
17. Read, J.: Scalable Multi-Label Classification. PhD thesis, University of Waikato, Hamilton, New Zealand (2010)
18. Demšar, J.: Statistical Comparisons of Classifiers over Multiple Data Sets. Journal of Machine Learning Research 7, 1–30 (2006)

Evolutionary Strategies Algorithm Based Approaches for the Linear Dynamic System Identification

Ivan Ryzhikov and Eugene Semenkin

Siberian State Aerospace University Named after Academician M. F. Reshetnev,
Krasnoyarsk, Russia
ryzhikov-88@yandex.ru

Abstract. In this paper the method of analytical form linear dynamic system identification is considered. The sample of the output measurements and the input control function are the only information that is required. The main problem was reduced to complex global optimization problem. Any solution that delivers an extremum to the criteria is a representation of a model structure and its parameters in the form of real numbers vector. The complexity of the reduced problem and its characteristics lead one to search for some special optimization technique. In the current research extremum seeking is based on a modification of the evolutionary strategies algorithm.

Keywords: differential equation, linear dynamic system, evolutionary strategies, identification.

1 Introduction

There are many different approaches for linear differential equation parameters estimation. But since there is no a priori information about the system itself and its structure, most of these approaches become useless or need a special treatment. For some tasks there are special techniques that were designed to solve the problem. In this paper we consider approach for the case when system's output observations can be noised and the structure is unknown. For given problem, for example, stochastic difference equations can be used, i.e. [1]. But there are some restrictions in using this approach: we still need the information about the order of differential equation and we must observe the system output on the unit step function. The specific problem of hysteresis identification is described in [2], where the implementing of dynamic neural network models sufficiently inreased efficiency oft he model. To simply estimate the reaction of linear dynamic system on different control input or smooth the output data we can use nonparametric methods, neural network of fuzzy output modelling. Also, there is a possibility to estimate the solution of differential equation for the described situation via genetic programming, [3]. As for nonparametric or neural network approaches it is possible to define the system output for different control function using the Cauchy equation, but the system cannot be presented in an analytical form. As for genetic programming technique we still have a possibility to find the output for different control, but since then, it can be found numerically. Moreover, the models are very complex and the analytical solution for estimation of differential

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 477–484, 2013.

equation seems to be very long and have a superior size. In this article seeking the model in differential equation form is suggested. The benefits are the following: it would be easy to estimate the system output numerically for any control function with any desired precision, in some cases it would be easy to define an analytical solution via eigenvalues evaluation, and there are plenty of control methods and analysis techniques for the models in differential equation form.

In article [4] the dynamic system approximation with second order linear differential equation via genetic algorithm is examined. The genetic algorithm is well known as effective global optimization technique. The only problem with it is that seeking works on a compact with given boarders and the real values ought to be quantized. In this paper we suggest to use an evolutionary strategies algorithm with local optimization and some modifications to approximate not only the parameters, but also the structure of an ordinary differential equation (ODE). Moreover, if the system's order is fixed one will get the best dynamic model with order less than the fixed value.

2 Identification of the Linear Dynamic System via Modified Evolutionary Strategies Algorithm

Let us have a sample $\{y_i, u_i, t_i\}, i = \overline{1,s}$, where s is its size, $y_i \in R$ are dynamic system output measurements at t_i and $u_i = u(t_i)$ are control measurements. It is also known, that the system is linear and dynamic, so it can be described with the ordinary differential equation (ODE):

$$a_k \cdot x^{(k)} + a_{k-1} \cdot x^{(k-1)} + \ldots + a_0 \cdot x = b \cdot u(t),\ x(0) = x_0. \quad (1)$$

Here the x_0 vector is supposed to be known. In the case of the transition process observing, one can put forward a hypothesis about the initial point values: the system output is known at initial time and the derivative values can be set to zero, because commonly the measurements of the system output begins while it is in a steady state. Generally, the initial point can be approximated or be a part of identification problem.

Using only the sample data it is necessary to identify parameters and the system order m, which is assumed to be limited, so $m \le M$, $M \in N$. This variable limits the ODE order. It is also assumed that there is an additive noise $\xi : E(\xi) = 0, D(\xi) < \infty$, that affects the output measurements:

$$y_i = x(t_i) + \xi_i. \quad (2)$$

Without information about the system order, we would not be able to solve the identification problm, but because of the maximum order limitation, the task can be partially parameterized. The maximum order is supposed to be chosen a priori. It would specify the optimization problem space dimension.

Without loss of the generality, let the leading coefficient of ODE be the constant and be equal to 1, so that

$$x^{(k)} + \tilde{a}_k \cdot x^{(k-1)} + \ldots + \tilde{a}_1 \cdot x = \tilde{b} \cdot u(t). \quad (3)$$

Then we can seek the solution of the identification task as a linear differential equation with the order m,

$$\hat{x}^{(m)} + \hat{a}_m \cdot \hat{x}^{(m-1)} + \ldots + \hat{a}_1 \cdot \hat{x} = \hat{a}_0 \cdot u(t),\ \hat{x}(0) = x_0, \tag{4}$$

where the vector of equation parameters $\hat{a} = (0, \ \ldots, \ 0, \ \hat{a}_m, \ \ldots, \ \hat{a}_1, \ \hat{a}_0)^T \in R^n$, $n = M + 1$, delivers an extremum to the functional

$$I(a) = \sum_{i=1}^{N} \left| y_i - \hat{x}(t_i) \right| \Big|_{\hat{a}=a} \to \min_{a \in R^n}. \tag{5}$$

In general case, the solution $\hat{x}(t)$ is evaluated with a numerical integration technique, because the control function has no analytical from, rather is given algorithmically. We prefer the criterion (5) instead of quadratic criteria, since absolute value of error is more robust in case of abnormal measurements than quadratic criteria. For the correct numerical scheme realization, let us have a coefficient restriction for equation (3), $|a_k| > 0.05$. Otherwise, this parameter is going to be equal to zero, and calculation scheme changes the parameters $a_k = 0, m = m - 1$. That condition prevents algorithm from extra computational efforts that would appear in the numerical evaluation scheme. Also, the given rule is necessary for the local optimization technique, since numerical optimization techniques commonly do not achieve the exact values of extremum point, but be in the local areas.

The reason why the main idea of optimization technique was borrowed from evolutionary strategies algorithm, [5] is that the identification problem leads to solution seeking for the multimodal optimization task. Moreover, there is no any information about the edges for the field of solutions, that is why, generally, optimization algorithms that works on the compact cannot satisfy our needs. The goal of the given approach is the identification of the parameters and the structure simultaneously. The system structure and its parameters are defined with one vector. The criterion (5) on the vector field is a complex mapping and it is sensitive to changes of the vector values, those are changing by stochastic search operators. That is the reason why it is important to implement the specific modifications for the global optimization technique, which could improve the searching efficiency.

Let every individual be represented with the tuple

$$H_i = \left\langle op^i, sp^i, fitness(op^i) \right\rangle, i = \overline{1, N_I},$$

where $op_j^i \in R,\ j = \overline{1,k}$ is the set of objective parameters of the differential equation, $sp_j^i \in R^+,\ j = \overline{1,k}$ is the set of strategic parameters, N_I is the population size, $fitness(x): R^k \to (0,1],\ fitness(x) = \dfrac{1}{1 + I(x)}$ is the fitness function.

Proportional, rank-based and tournament-based selections were implemented. The algorithm produces one offspring from two parents and every next population have the same size as previous. Recombination types are intermediate and discrete.

The mutation of every offspring's gene takes place with the chosen probability p_m. The random value $z = \{0,1\}, P(z=1) = p_m$, which is generated for every current objective gene and its strategic parameter, make it possible to control the level of mutation

$$op_i^{offspring} = op_i^{offspring} + z \cdot N(0, sp_i^{offspring});$$
$$sp_i^{offspring} = \left| sp_i^{offspring} + z \cdot N(0,1) \right|,$$

where $N(m, \sigma^2)$ is the normally distributed random value with the mean m and the variance σ^2.

We suggest a new operation that could increase the efficiency of the given algorithm. For every individual, the real value is rounded down to the nearest integer. This provides searching for solutions with near the same structure.

Also for N_1 randomly chosen individuals and for N_2 randomly chosen objective chromosomes we make N_3 iterations of local search with the step h_l to determine the better solution. Current technique is the random coordinate-wise optimization.

There was an investigation of efficiency for different optimization algorithms with modifications that are described below. For the investigation 100 systems were generated: 10 for every order from first to tenth. Parameters of the systems were randomly generated: $\hat{a}_k^i = U(-5,5)$, $\hat{b} = U(-5,5)$, $i = \overline{1,10}$, $k = \overline{1,i}$, where $U(-5,5)$ is generated uniformly distributed value. The time of the process was set to 5. The control function was the step function and we know what was the control for every system, so $u(t) = 1$. Let $\{x_i, t_i\}, i = \overline{1, T/h_i}$ be the numerical solution for the differential equation. We take $s < T/h_i, s = 100$ points randomly. For every system 10 runs of the algorithm were executed with every combination of its parameters. To estimate the efficiency of different approaches we considered the identification without any noise. Having different types of the selection and the crossover, we would also vary the p_m to find out the most effective combination of the algorithm settings. As a pre-set we use the population size in 50, the number of populations in 50, $N_1 = 50$, $N_2 = 50$ and $N_3 = 1$ with $h_l = 0.05$.

We compared the efficiency of following algorithms: 1 – the evolutionary strategies (ES) algorithm; 2 – ES with the local optimization, hybrid evolutionary strategies (HES); 3 – HES with modified mutation; 4 – HES with turning real numbers into integer numbers; 5 - HES with modified mutation and turning real numbers to integer ones. After testing the algorithms on different samples effective settings were found.

For the proper structure and parameters estimation we need an adequate sample that reflects all the transient process. Let us take some stable systems that come into the steady state in time $T = 5$. There is an efficiency investigation for the modified HES algorithm that shows the relation between the fitness function and identifying probability of exactly the same structure and parameters as differential equation has. 20 runs of the algorithm were made for every system. We will say that the algorithm determines the structure and parameters if $\max(\hat{a} - a) < 0.05$. The earlier investigation results proved that the fitness function is not correlated with probability of identifying exactly the same parameters and structure. On the figure 1 the efficiency of different optimization algorithms are presented for the testing sample – 100 samples.

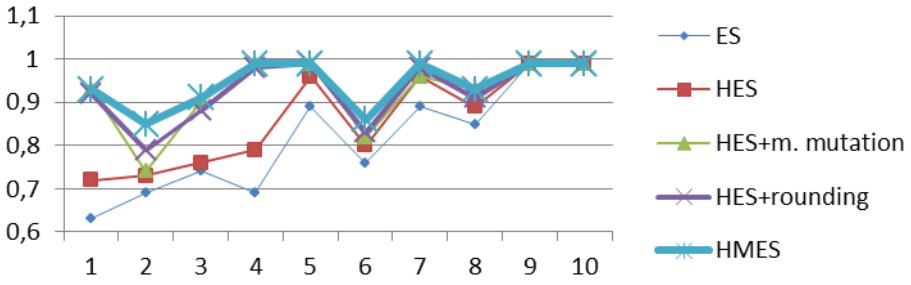


Fig. 1. The means of fitness function value for different orders and different optimization techniques

Let us highlight that for many solutions which were found for stable systems, the order was found correctly. Actually, the miss in the real parameters identification does not mean that the system was identified completely wrong. The fact that fitness function is close to 1 means that the transient process itself is found correctly so the solution for identification problem is adequate and could be useful for control problems at least if the time interval is the same as the time the system was observed.

Without the operation of rounding the parameters the different evolutionary algorithm could be used simultaneously in a following way: every distinct evolutionary strategies algorithm is to solve the identification problem only for model with fixed order of differential equation. So the whole identification scheme works as brute force for every order and it uses stochastic search for its parameters. For the case when the system order is known the efficient parameters of the optimization algorithm were estimated and used in the general scheme. It is important to highlight that the numerical resources were nearly the same as with any of the previously described algorithms. Since the algorithms work on vector fields with different dimensions, the increasing only of the local optimization steps number were used to equalize the efficiency. Managing the amount of resources for local optimization is the easiest and the most effective way to keep the quality of found solutions. Actually, there could be different ways to control the efficiency in case of the problem dimension increasing, but more detailed research is the main part of the further investigation. On figures 2 and 3 the comparements on the algorithm schemes are demonstrated.

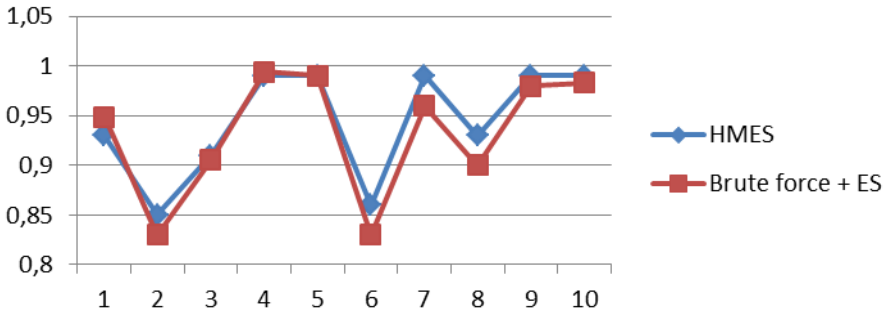


Fig. 2. The means of fitness function value for HMES and Brute force – based algorithms

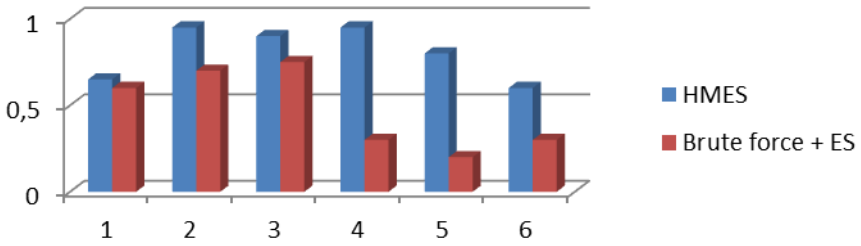


Fig. 3. The proper structure and parameters identification probability estimation

As it is shown on the figures, the mean of fitness function are slightly different, but the quality of the identification problem solutions is not the same. In the further investigation the testing sample would be increased and the new statistical characteristics would be under consideration.

It is more likely to be true that with the same computational resources algorithms are not equal. To find the reason and a way to improve the multi-ES scheme is a very important problem.

3 Hexadecane Disintegration Reaction Identification

Let us describe the identification problem for hexadecane chemical reaction. The disintegration of the hexadecane gives the following products: the spirits and carbonyl compounds. The initial point is known. There is no control input in this identification problem. We set the maximum order for the first equation to 10. The 50 runs of the algorithm gave us some different solutions that are shown in Table 1.

As we can see, the found parameters and system structure forms the first order differential equation, and that fact does not contradict the hypothesis [6], which states that disintegration chemical reactions can be presented as first order linear differential equation.

Knowing the structure of the equations we can identify the system itself in a matrix form. The given optimization procedure is a stochastic algorithm, that is why the best solution from the 20 runs was taken. The system outputs and the sample are shown on figure 4 for hexadecane, spirits and carbonyl compounds. As it is shown on figure, the measurement at the point $t = 7$ seems to be an abnormal measurement, but it did not effect on the model.

In earlier researches the ODE model for the disintegration process was found. Now we put forward a hypothesis about existing of free coefficient in every equation in the system. The modified evolutionary strategies algorithm without rounding after 20 restarts for the identification of parameters for the system of linear differential equations found the following solution:

$$x' = \begin{pmatrix} -0.2136 & 0.8448 & -0.5420 \\ -0.0899 & -0.4785 & -0.0545 \\ 0.0772 & 0.7317 & -0.4720 \end{pmatrix} \cdot x + \begin{pmatrix} 0.5795 \\ 0.7155 \\ -0.4408 \end{pmatrix}.$$

Table 1. The hexadecane disintegration model

Model	The error
$4.05 \cdot x' + 0.9 \cdot x = 1,$	$I = 0.3022$
$1.05 \cdot x' + 0.4 \cdot x = 1,$	$I = 0.2834$
$2.1 \cdot x' + 0.55 \cdot x = 1,$	$I = 0.1822$
$-1.05 \cdot x''' - 0.15 \cdot x'' - 6.85 \cdot x' - 0.9 \cdot x = 1,$	$I = 0.227$
$-3.4 \cdot x' - 0.45 \cdot x = 0.$	$I = 0.202$

The criterion value lessen by 3% by adding the free coefficients. On the figure 4 the model output and the sample are presented. As it can be seen the abnormal measurement at the point $t = 7$ did no effect to the model.

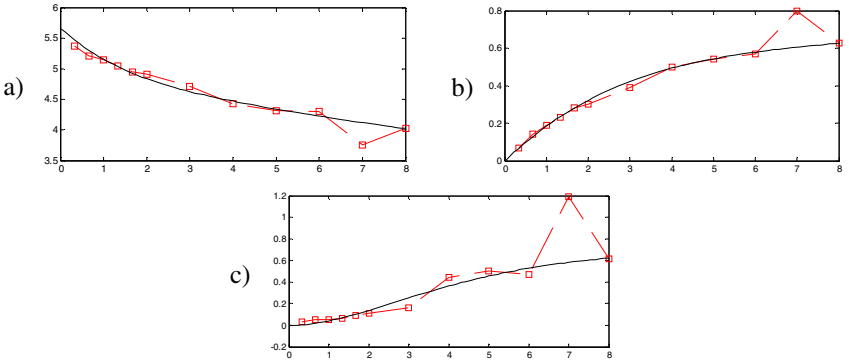


Fig. 4. Hexadecane (a), spirits (b) and carbonyl compounds (c) concentration measurements and model output, respectively.

Modifications of evolutionary strategies algorithm increased the accuracy of model and allowed solving two tasks at the same time: the system order and parameters estimation. The further investigation focuses on comparing different evolution-based algorithms for the linear dynamic system identification problem. Effective settings estimation and the recommendations how to choose ones is an important problem for identification and optimization needs in case of using different algorithms for different structures of differential equations.

References

1. Zoteev, V.: Parametrical identification of linear dynamical system on the basis of stochastic difference equations. Matem. Mod. 20(9), 120–128 (2008)
2. Tan, Y., Dong, R., Chien, H., He, H.: Neural networks based identification of hysteresis in human meridian systems. Int. Journal of Applied Mathematics and Computer Science 22(3), 685–694 (2012)

3. Cao, H., Kang, L., Chen, Y., Yu, J.: Evolutionary modeling of systems of ordinary differential equations with genetic programming. Genetic Programming and Evolvable Machine 1(40), 309–337 (2000)
4. Parmar, G., Prasad, R., Mukherjee, S.: Order reduction of linear dynamic systems using stability equation method and GA. International Journal of Computer and Infornation Engeneering 1(1), 26–32 (2007)
5. Hans-Paul, S.: Evolution and Optimum Seeking. Wiley & Sons, New York (1995)
6. Romanovskii, B.V.: The foundations of the chemical kinetics. Ekzamen, Moscow (1996) (in Russian)

A Genetic Algorithm Approach for Minimizing the Number of Columnar Runs in a Column Store Table

Jane Jovanovski, Maja Siljanoska, and Goran Velinov

Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius University, Skopje, Macedonia
{janejovanovski,maja.siljanoska}@gmail.com,
goran.velinov@finki.ukim.mk

Abstract. Column-oriented database systems, usually referred to as column stores, organize data in a column-wise manner. Column-wise data can be compressed efficiently, improving the performance of large read-mostly data repositories such as data warehouses. Many compression algorithms exploit the similarity among the column values, where repeats of the same value form columnar runs. In this paper we present a genetic algorithm for determining an optimal column sorting order which will minimize the number of columnar runs in a column store table and therefore maximize the RLE-based table compression. Experiments show that the algorithm performs consistently well on synthetic table instances as well as realistic datasets, resulting with higher run-reduction efficiency compared to existing heuristic for solving the given problem.

Keywords: column stores, RLE-based compression, columnar runs, run-reduction, genetic algorithms, permutation representation.

1 Introduction

Traditional database systems use row-oriented data storage where values from different columns of a record (row) are stored together. This enables high performance writes which is especially beneficial for OLTP applications. However, it does not work well with systems, such as data warehouses, which are oriented towards ad-hoc querying of large amounts of data. In this case, better performance is achieved by using a column-wise data organization [2].

In column-oriented database systems, referred to as column stores, values from a single column in different records are stored contiguously, typically densely packed, whereas the traditional database systems store entire records one after another [1]. This reduces the data processed by a query because it reads only the columns it needs. Column-oriented data are highly amenable to compression, enabling column stores to optimize their storage space and utilize the storage optimization to improve their performance for a read-mostly query workload.

Storing data by columns greatly increases the similarity between adjacent column values, which enhances the compressibility of the data [3]. Many compression algorithms exploit this similarity by minimizing the number of columnar

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 485–494, 2013.

runs, that is, the number of repeats of the same value in each column. There are two general approaches to minimize the number of columnar runs. The first is minimization by row reordering, which refers to rearranging the rows in such a way that the number of columnar runs in the table is as minimal as possible. The second is minimization by table sorting, based on the idea that sorting the table improves compression, and permuting the columns in the right order before sorting can reduce the number of columnar runs by a factor of two or more. The problem of determining an optimal row reordering and the problem of determining an optimal column sorting order are both known to be NP-hard [12]. This makes them good candidates for applying metaheuristic optimization methods such as genetic algorithms.

In this paper we address the problem of minimizing the number of columnar runs by sorting a column store table in an optimal column sorting order. An often recommended heuristic for solving this problem is by lexicographic sorting with "low cardinality columns serving as the leftmost sort orders" [3]. This empirical recommendation is justified and the heuristic is proven to give good results [12]. We develop a genetic algorithm to solve the given problem and demonstrate that this approach produces better quality solutions for experiments performed on realistic datasets. Our algorithm shows improved run-reduction efficiency up to a factor of 4 compared to the reduction achieved by the given heuristic [12].

The paper is organized as follows. Section 2 defines the problem of minimizing the number of columnar runs by using table sorting to achieve the minimization. Section 3 describes the design and implementation details of the proposed genetic algorithm. In Section 4 we present the experiments made both on synthetic data and realistic datasets and analyze the obtained results. Finally, we give some conclusions and directions for future work in Section 5.

2 Problem Definition

One advantage of column stores is data compression which helps to reduce storage space as well as I/O times. An attractive approach for compressing sorted data in a column store is the run-length encoding (RLE) where repeats (runs) of the same value are stored as a single data value and count. RLE performs lossless compression: the original data can be reconstructed from the compressed data.

Columns in column stores can be compressed by using the repetition and similarity among values within a column, where sequences of adjacent column cells with identical values form columnar runs. Each columnar run is stored as an RLE pair (value, run-length). Accessing a random position in this kind of data structure requires an $O(r)$ operation for a column with r runs. The search bounds can be improved to $O(log_2(r))$ by enabling binary search. This is achieved by using redundant data: each columnar run can be represented as an RLE triple (value, start-position, run-length) or (value, start-position, end-position). Storing the columnar runs as RLE triples ensures preservation of the original row structure and eases the process of table reconstruction.

RLE-based table compression is more efficient for tables with fewer columnar runs. Thus the number of columnar runs can be used as a general model for

RLE-based table compression, i.e. the problem of RLE-based compression of column store tables can be viewed as the problem of minimizing the number of columnar runs defined as follows: Given a column store table with M columns, find the optimal column sorting order which minimizes the total number of columnar runs $\sum_{i=1}^{M} r_i$, where r_i is the number of columnar runs in the i-th column, $i = 1, 2, ..., M$.

For example, Table 1 represents the table Customer whose total number of columnar runs is 23, however this number is not optimal. Table 2 shows the same table after sorting its columns in one possible optimal order when the number of columnar runs is minimized and is 15, whereas Table 3 displays the corresponding RLE-based compression of the table.

Table 1. The original Customer table (before sorting)

Row	1	2	3	4
Col	Name	City	Color	Gender
1	Nick	Amsterdam	Blue	M
2	Mary	Zurich	Purple	F
3	Tom	Rome	Blue	M
4	Nick	London	White	M
5	Cris	Zurich	Blue	F
6	Tomas	Amsterdam	White	M

Table 2. The Customer table after sorting in an optimal column order

Row	1	2	3	4
Col	Name	City	Color	Gender
1	Nick	Amsterdam	Blue	M
2	Nick	London	White	M
3	Tom	Amsterdam	White	M
4	Tomas	Rome	Blue	M
5	Cris	Zurich	Blue	F
6	Mary	Zurich	Purple	F

Table 3. The Customer table compressed using RLE-based compression

Row	1	2	3	4
Col	Name	City	Color	Gender
1	Nick[1,2]	Amsterdam[1,1]	Blue[1,1]	M[1,4]
2	Tomas[3,4]	London[2,2]	White[2,3]	F[5,6]
3	Cris[5,5]	Amsterdam[3,3]	Blue[4,5]	
4	Mary[6,6]	Rome[4,4]	Purple[6,6]	
5		Zurich[5,6]		
6				

3 The Genetic Algorithm

3.1 Representation of the Solution Domain

The problem of finding an optimal column sorting order for a table with M columns consists of determining a permutation of the non-repeating sequence $1, 2, ..., M$ that represents the order in which the columns are sorted such that the total number of columnar runs in the table is minimized. Each permutation element is assigned a flag value to note the order of sorting (ascending, descending or none).

Therefore, the chromosomes that represent candidate solutions of the problem consist of an ordered set $\{(c_i, s_i) \,|i = 1, 2, ..., M\}$ where $\{c_1, c_2, ..., c_M\}$ is a permutation on the set $\{1, 2, ..., M\}$ representing the order in which the M columns of the table are sorted, and $\{s_1, s_2, ..., s_M\}$ is a permutation with repetition of M elements on the set $\{0, 1, 2\}$ denoting no sorting, ascending and descending sorting respectively. Hence the search space of the problem contains in total $M! \cdot 3^M$ possible chromosomes.

For example, the chromosome $\{(4, 2), (1, 1), (2, 1), (3, 0)\}$ represents one optimal column sorting order for Table 2. Initially the sorting is done by the 4th column (Gender) in decreasing order; table rows with same values for Gender are then sorted by column 1 (Name) in ascending order; finally rows with same values for Name are sorted in ascending order by column 2 (City). This corresponds to the following SQL ORDER BY clause:

```
SELECT *
FROM Customer
ORDER BY Gender DESC, Name ASC, City ASC
```

3.2 Fitness Function

In the given problem, the objective function is the total number of columnar runs in a table. If the table has M columns, the fitness function is defined as:

$$f(x) = \sum_{i=1}^{M} r_i(x) \ ,$$

where $r_i(x)$ gives the number of runs of identical value in the i-th column for a column sorting order corresponding to a chromosome x, for $i = 1, 2, ..., M$.

3.3 Initial Population

We implemented two approaches for generation of the initial population.

Random Generation. For a population size of K_{pop} chromosomes, the initial population is generated using the following procedure. $K_{pop} - 1$ chromosomes are chosen randomly as follows: for each chromosome, a random permutation of M elements is generated using the Knuth shuffle algorithm [10] (any permutation of M elements will be produced with probability of exactly $1/M!$, thus yielding a uniform distribution over all such permutations). Then, for each of the elements in the permutation a randomly selected element from the set $\{0, 1, 2\}$ is assigned. The chromosome $\{(1, 0), (2, 0), ..., (M, 0)\}$ formed by the identity permutation which corresponds to the original table is added to the initial population, to ensure that the initial population's best fitness value is at least as good as the fitness value of the original table and that no good results are being lost.

Heuristic-Based Generation. The cardinality of a column denotes the number of distinct values in the column. There are three types of cardinality related

to columnar value sets: low cardinality which refers to columns with few unique values (e.g. status flags, boolean values or major classifications), normal cardinality referring to columns with values that are uncommon but never unique (e.g. names or street addresses), and high cardinality which refers to columns with values that are very uncommon or unique (e.g. id numbers, emails or usernames). To formalize these notions, let $R_1, R_2 \in (0,1)$, $R_1 \leq R_2$. Given a table with N rows, a column with cardinality r is said to have low cardinality if $\frac{r}{N} \in (0, R_1]$ and high cardinality if $\frac{r}{N} \in (R_2, 1]$. Otherwise, it will have normal cardinality.

Columns with low (high) cardinality are likely to form longer (shorter) runs of identical values and produce smaller (greater) number of runs upon sorting. Following this reasoning, a heuristic based on column cardinalities can be employed to seed the initial population as follows. One chromosome represents the original table and the other $K_{pop} - 1$ chromosomes are generated as in the first approach, however the sorting flag value is defined by applying the following heuristic: low cardinality columns are always considered for sorting and their corresponding permutation elements are assigned a flag value from the subset $\{1, 2\}$, whereas high cardinality columns are not sorted (flag value 0). Columns with normal cardinality are randomly assigned any value from the set $\{0, 1, 2\}$.

3.4 Genetic Operators

The design choices for the genetic operators are briefly described below.

Selection. The selection of individuals for the mating pool is performed by exploiting the tournament selection method, implemented by holding a tournament between $k \geq 2$ randomly chosen individuals. This algorithm is computationally more efficient (no sorting is required) and more amenable to parallel implementation compared to other selection algorithms [14].

Crossover. Our genetic algorithm implements the three most simple and commonly used crossover operators which ensure permutation chromosome feasibility: Cycle Crossover (CX), Order Crossover (OX) and Partially Matched Crossover (PMX) [16,11]. CX performs a recombination under the constraint that each gene comes from one of the parents. OX constructs an offspring by choosing a subsequence of one parent and preserving the relative order of the genes of the other parent, while PMX maps a portion of one parent's genes into a portion of the other parent's genes and exchanges the remaining information. Crossover is applied with a probability P_c, hence for a population of K_{pop} chromosomes, $\lceil P_c \cdot K_{pop} \rceil$ offspring chromosomes will be produced with crossover.

Mutation. If the population size is K_{pop} and the mutation probability is P_m, then $\lfloor P_m \cdot K_{pop} \rfloor$ offspring chromosomes will be subject of mutation. Three mutation operators that preserve the ordering property of permutation chromosomes were implemented: Insertion Mutation (IM), Simple Inversion Mutation (SIM) and Swap Mutation (SM) [16,11]. IM removes a randomly chosen gene and reinserts it in a random location in the chromosome. SIM reverses the chromosome section between two randomly chosen cut points, while SM exchanges (swaps) the content of two randomly selected genes in the chromosome.

Replacement. Our algorithm uses generational updates of the population as well as elitism strategy to improve the performance. Namely, $\lfloor (1 - P_c) \cdot K_{pop} \rfloor$ of the best individuals of the parent population are retained in the new population, while the remaining $\lceil P_c \cdot K_{pop} \rceil$ chromosomes in the new population are produced by crossover from chromosomes in the parent population. This ensures that the best individuals from each population are not lost or destroyed [14] and significantly improves the algorithm's performance.

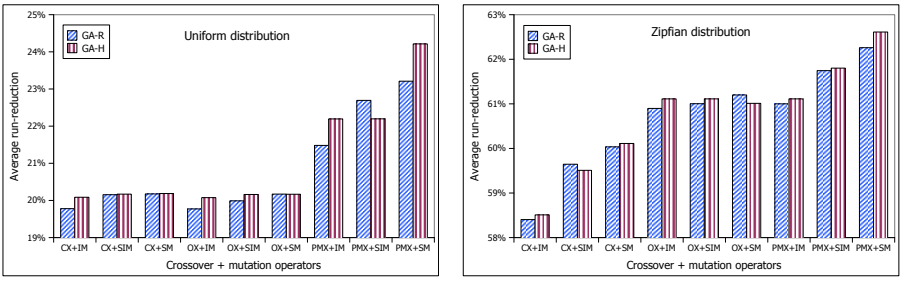
4 Experimental Results

We present two groups of experiments with the purpose of evaluating the performance of the proposed genetic algorithm approaches. First, we use synthetic data to compare and assess GA-R (the GA using randomly generated initial population) and GA-H (the GA using heuristic for seeding the initial population) for different combinations of the crossover and mutation operators in order to find the one that gives best results. Then we exploit these results to demonstrate that our genetic algorithm performs better than the existing heuristic H-LK [12]. Eventually we show that the theoretical results obtained on synthetic data are appealing and applicable on realistic datasets as well.

All experiments were performed using the following parameters which proved to be robust in preliminary tests. The process of optimization was run using a crossover probability of 90 % ($P_c = 0.9$) and a mutation rate of 5 % ($P_m = 0.05$), over a population of 100 chromosomes ($K_{pop} = 100$). The tournament size in the tournament selection was chosen to be $k = 2$. GA-H was employed with cardinality parameters $R_1 = 0.1$ and $R_2 = 0.9$. For each experiment 10 independent trials were performed and the standard deviation of the resulting data was observed in order to examine the stability of the algorithm. The overall performance measure used for comparisons was the run-reduction percentage averaged over the 10 independent trials.

4.1 Experiments with Synthetic Data

For uniformly distributed tables with M columns whose cardinalities are r_1, r_2, ..., r_M, any value within column i can take one of r_i distinct values with probability $1/r_i$, for $i = \overline{1, M}$. We generated 16 such tables, with 1000, 10000, 30000 and 50000 rows, using 10, 15, 20 and 25 independently generated columns with uniform distribution of their values. Our first objective was to find which combination of crossover and mutation operators gives best quality solutions for GA-R and GA-H and which of these two genetic algorithms performs better. For that purpose, both GA-R and GA-H were run over the 16 uniformly distributed tables for each of the 9 combinations of crossover (CX, OX, PMX) and mutation (IM, SIM, SM) operators, yielding 288 different experiments with 10 independent trials for each experiment, for a total of 2880 experimental trials. The maximum deviation value of the obtained data was observed to be 3.9 %. Fig. 1(a) shows the averaged reduction of the number of columnar runs achieved by GA-R and



(a) Uniformly-distributed case (b) Zipfian-distributed case

Fig. 1. Average run-reduction of GA-R and GA-H vs. crossover and mutation

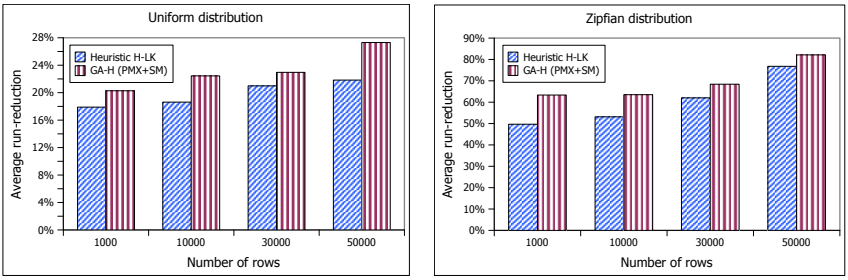
GA-H for each combination of crossover and mutation operators. As it can be seen from the figure, GA-H performs better on average than GA-R with highest reduction achieved for PMX and SM operators in 81.25 % of the experiments.

These results assume uniformity, thus there is a need to assess the reliability of the obtained results also for skewed data. The Zipfian distribution and its modifications are commonly used to model value distributions in databases [5,9]. The frequency of the i-th value within a column is proportional to $1/i$. If the table has N rows, then each column can have N possible distinct values, however not all of them will normally appear. Following this, we generated 16 Zipfian-distributed tables for the same number of rows and columns as in the uniformly distributed case. Each table column was generated independently with the skeweness parameter varying between $z = 0.2$ (low skew) and $z = 2.0$ (high skew). These tables were subject to 288 experiments, with 10 independent trials each, observing maximum standard deviation of 4.2 % for the obtained experimental data. The averaged run-reduction results obtained by GA-R and GA-H for the different crossover and mutation operators is provided in Fig. 1(b). These results show that GA-H again outperforms GA-R for the combination of PMX and SM operators, giving highest reduction in 75 % of the experiments.

Generalizing the results jointly for all experiments performed on uniformly and Zipfian-distributed tables, GA-H (PMX+SM) has best reduction performance in 78.13 % of the cases compared to the other genetic algorithm variants that were taken into consideration. Therefore, GA-H (PMX+SM) will be used as our genetic algorithm of choice in all further experimental applications.

Following these analytical results, our next aim is to compare the performance of GA-H (PMX+SM) with the performance of the heuristic H-LK. We performed the comparison both for uniform and skewed data, using the tables generated before. The results from the comparative analysis of the average run-reduction per number of rows for the uniformly distributed tables are presented in Fig. 2(a) and for the Zipfian-distributed tables in Fig. 2(b). This shows that the run-reduction efficiency grows with the number of rows both for uniform and Zipfian-distributed tables. Furthermore, the results obtained demonstrate that in 100 % of the experiments GA-H achieves better results compared to H-LK for uniform data, with a maximum reduction gain of 5.47 % and 3.89 % on average. Similar

findings hold for Zipfian-distributed data as well: GA-H always results with higher run-reduction compared to H-LK. This reduction difference in the Zipfian case is 8.05 % on average, with maximum reduction of 12.04 %. Here we see improved gains compared to the uniform model because a Zipfian model implies more opportunities to create long runs of identical values in several columns.



(a) Uniformly-distributed case (b) Zipfian-distributed case

Fig. 2. Average run-reduction of H-LK and GA-H per number of rows

4.2 Experiments with Realistic Datasets

We demonstrated above that our genetic algorithm GA-H (using PMX and SM operators) outperforms the existing heuristic H-LK in terms of higher reduction of the number of columnar runs when applied on uniform and skewed synthetic data. In order to assess the reliability and applicability of this assertion, we performed some experiments on realistic datasets as well. We used five publicly available datasets representative of real-life data tables: BCUMB and CUMB1881 [6], Census-Income [8], and Nursery and Poker-Hand [7]. BCUMB contains records about the birth registrations in Cumberland County. CUMB1881 represents records from the Cumberland County Census 1881. Census-Income contains weighted census data extracted from the 1994 and 1995 current population surveys conducted by the U.S. Census Bureau. Nursery dataset was derived from a hierarchical decision model originally developed to rank applications for nursery schools in Slovenia [15]. In Poker-Hand each record is an example of a hand consisting of five playing cards drawn from a standard deck of 52 [4]. The characteristics of these datasets are given in Table 4. The parameter ρ_0 is a simple measure of the statistical dispersion of the frequency of the table values [13], and for N rows and M columns is computed as $\rho_0 = \sum_{i=0}^{M} \frac{f(v_i)}{NM}$, where $f(v_i)$ is the frequency of the most frequent value v_i within a column i, for $i = \overline{1, M}$.

The experimental results are summed in Table 5. They demonstrate that our genetic algorithm achieves higher run-reduction compared to H-LK in all cases. The reduction improvements are up to 24.6 % in the best case. Most noteworthy are the improvements for the CUMB1881 dataset with a reduction of nearly factor 4 (33.13 % vs 8.53 %) and for the BCUMB dataset with a reduction of nearly a factor 2 (12.42 % vs 6.57 %).

Table 4. Characteristics of the realistic datasets used in the experiments

	rows	columns	min. card.	max. card.	ρ_0
BCUMB	5953	20	34	1357	0.12
Nursery	12960	9	2	6	0.32
CUMB1881	27363	9	22	5018	0.06
Census-Income	199523	42	2	99800	0.65
Poker-Hand	1025010	11	4	13	0.19

Table 5. Performance of GA-H (PMX+SM) and H-LK on realistic datasets

	Initial fitness	Heuristic H-LK	GA-H (PMX+SM)
BCUMB	80880	75567 (6.57 %)	70835 (12.42 %)
Nursery	29617	19439 (34.37 %)	17668 (40.35 %)
CUMB1881	122519	112071 (8.53 %)	81931 (33.13 %)
Census-Income	3764757	1053819 (72.01 %)	947872 (74.82 %)
Poker-Hand	9157488	3179577 (65.28 %)	3148680 (65.62 %)

All these experiments performed both on synthetic data and realistic datasets ascertain that our genetic algorithm approach is better in terms of quality of the obtained solutions compared to the existing heuristic. However, it is fair to mention that the GA approach is more computationally expensive due to the fact that evaluating each candidate solution implies sorting of the table, unlike the existing heuristic which considers only a few easily computed statistics such as column cardinality. Nevertheless, in the case of data warehouses, any compression gain can be useful regardless of the computational time because the compression is done once and is used as such afterwards.

5 Conclusions and Future Work

In this paper we presented a generational genetic algorithm for determining an optimal column sorting order which minimizes the number of columnar runs in a column store table. The algorithm employs permutation representation, pairwise tournament selection, elitism, and three different crossover and mutation operators. It was implemented and tested using both random and column cardinalities-based heuristic generation of the initial population. The experiments performed on uniform and skewed data revealed that best results are achieved when using heuristic-based initial population with PMX crossover and SM mutation. Moreover, it was shown that this algorithm performs consistently well both on synthetic data and on realistic datasets and can achieve up to 4 times higher reduction compared to existing heuristic for solving the given problem.

Future work is currently underway to investigate the possibility of incorporating the H-LK heuristic in the initial population and some problem-specific variation operators which could possibly make the algorithm more powerful. Furthermore, the main advantage of any genetic algorithm lies in its intrinsic parallelism nature and therefore different island models for parallelization should

be implemented and tested. It could be also interesting to mine the results from experiments conducted on various data distributions in order to develop an analytical decision support tool which will help us to associate given data with the most appropriate approach to minimize the number of columnar runs.

References

1. Abadi, D.J., Boncz, P.A., Harizopoulos, S.: Column-oriented Database Systems. Proceedings of the VLDB Endowment 2(2), 1664–1665 (2009)
2. Abadi, D.J., Madden, S.R., Hachem, N.: Column-Stores vs. Row Stores: How Different Are They Really? In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Vancouver, Canada, pp. 967–980 (2008)
3. Abadi, D.J., Madden, S.R., Ferreira, M.C.: Integrating Compression and Execution in Column-Oriented Database Systems. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, USA, pp. 671–682 (2006)
4. Cattral, R., Oppacher, F., Deugo, D.: Evolutionary Data Mining with Automatic Rule Generalization. In: Recent Advances in Computers, Computing and Communications, pp. 296–300. WSEAS Press (2002)
5. Eavis, T., Cueva, D.: A Hilbert Space Compression Architecture for Data Warehouse Environments. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2007. LNCS, vol. 4654, pp. 1–12. Springer, Heidelberg (2007)
6. Edwards, G.: Nova Scotia GenWeb Project, Cumberland County GenWeb, `http://www.rootsweb.ancestry.com/~nscumber/sources.html` (checked October 20, 2012)
7. Frank, A., Asuncion, A.: UCI Machine Learning Repository (2010), `http://archive.ics.uci.edu/ml` (checked October 20, 2012)
8. Hettich, S., Bay, S.D.: The UCI KDD archive (2000), `http://kdd.ics.uci.edu` (checked October 20, 2012)
9. Houkjær, L., Torp, K., Wind, K.: Simple and realistic data generation. In: Proceedings of the 32nd International Conference on Very Large Data Bases, Seoul, Korea, pp. 1243–1246 (2006)
10. Knuth, D.E.: The Art of Computer Programming, vol. 2, pp. 124–125. Addison-Wesley (1969)
11. Lee, K.Y., El-Sharkawi, M.A.: Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems. Wiley-IEEE Press (2008)
12. Lemire, L., Kaser, O.: Reordering Columns for Smaller Indexes. International Journal of Information Sciences 181(12), 2550–2570 (2011)
13. Lemire, L., Kaser, O., Gutarra, E.: Reordering Rows for Better Compression: Beyond the Lexicographic Order. ACM Transactions on Database Systems 37(3), 20 (2012)
14. Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, Cambridge (1998)
15. Olave, M., Rajkovic, V., Bohanec, M.: An application for admission in public school systems. In: Expert Systems in Public Administration, pp. 145–160. Elsevier Science Publishers (1989)
16. Sivanandam, S.N., Deepa, S.N.: Introduction to Genetic Algorithms. Springer, Heidelberg (2008)

Shadow Detection in Complex Images Using Neural Networks: Application to Wine Grape Seed Segmentation

Felipe Avila[1], Marco Mora[1], Claudio Fredes[2], and Paulo Gonzalez[1]

[1] Department of Computer Science, Universidad Católica del Maule, Chile
favila.ici@gmail.com, mora@spock.ucm.cl, pgonzalezg@ucm.cl
http://www.ganimides.ucm.cl/mmora
[2] Department of Agronomy, Universidad Católica del Maule, Chile
cfredes@ucm.cl

Abstract. Determining the exact point of ripening and harvesting of the grapes is essential for obtaining a wine of quality. Recent methods for determining the ripening of the grapes are based on visual inspection of the seed. These methods have the advantage of being simple and of low-cost, but they are prone to human error, and a large number of samples are required to be analyzed in order to obtain representative information of the reality. Currently, the analysis of the seed is made using images obtained with a digital camera, which have major problems as the existence of shadows and highlights. This paper proposes a segmentation method of grape seed in complex images based on artificial neural networks and color images. The method is robust to imperfections in the images, which permits that this type of analysis is installed in reality.

Keywords: Shadow Detection, Wine Grape Seed Segmentation, Neural Networks.

1 Introduction

Wine is a highly valued beverage because of its natural origin. In wine-producing countries, to improve its production process has economic and social relevance. One of the key factors for a quality wine is to correctly determine the time at which the grapes must be harvested. Traditionally the estimation of optimum ripeness is performed by a human expert (wine expert) that analyzes the color and flavor of the grape, or by laboratory chemical analysis. Methods to determine ripeness based on visual inspection of the seed have been proposed recently [1], showing a high correlation between color, texture and shape of the seed with respect to the ripeness state of the grapes.

Recent work in the line of the inspection of the seed is in [2], where it is proposed to compare the color of the seed skin on a color scale developed in the same work. The proposed methodology is simple and low-cost, so its implementation in reality is of interest. Visual inspection of the seed is very susceptible to human error, in the sense that color perception is not always correct. Moreover,

M. Tomassini et al. (Eds.): ICANNGA 2013, LNCS 7824, pp. 495–503, 2013.

a significant amount of seeds must be analyzed so that the findings of the test are relevant. To address the above problems, this paper intends to make visual inspection of the seed as a Pattern Recognition problem in color images. Digital images of the seed are obtained, and then segmentation and later classification of the seed in types of ripeness of interest for the expert. In particular this research addresses the segmentation stage of the grape seed.

The images discussed in this paper are those acquired in [2]. These images have various problems such as the presence of projected shadow [3], and low contrast between the object and shadow. The proper removal of shadows is a difficult task and it is relevant for proper segmentation of objects in the image. For the classification of shadows there is a strong trend which consists of applying of color models invariant to lighting. An initial work with this approach is in [4], where it is proposed to detect and classify shadows for a static image. Detection of edges and morphological operators are used to extract the regions of shadow. In [5] the previous work is extended to motion images. In [6] a method to remove shadows based on a representation of the color image using an invariant model in grayscale is proposed. Edges are detected in both the invariant and the original image, and those edge pixels that are in the original image but not present in the invariant are considered as edge of the shadow. The problem with this method is that to generate the invariant it is necessary to know the angle of projection of the shadow, which can be very complex. In [7] a technique that combines the use of two invariant models is proposed, the first one is based on a normalization of the RGB and the second is the model proposed in [6]. Methods based on invariant to lightning are interesting because they are simple to implement. However, in very complex images such as grape seed good results are not obtained. In this paper, an improvement to the methods that use invariants to detect shadows based on Neural Networks is proposed. The object is segmented through supervised learning and through the invariant, and both results are combined to yield substantial improvements in the implementation of the methods separately.

The experiments of this work were done in Matlab, Toolbox Balu for Pattern Recognition was particularly used[1]. The structure of the paper is as follows: section 2 presents details of the proposal, section 3 shows the results of the method, and finally section 4 outlines the conclusions of the study and future works arising from this research.

2 Hybrid Method for Shadow Detection

The focus of this section is to describe in detail the hybrid method for detecting shadows. This method combines an unsupervised approach based on color models invariant to lightning and a supervised approach based on neural networks.

[1] Toolbox available from `http://dmery.ing.puc.cl/index.php/balu/`

2.1 Identification of Relevant Features for Classification

A first approach to separate the seed and the shadow in the image was to study the characteristics of color. For this purpose, samples of pixels of both seed and shadow in different images. The values of these pixels were transformed into various color models (RGB, HSV, YIQ, YCbCr, XYZ, CMYK, Lab, Luv, etc.), including invariant models proposed in [8]. To determine which channels allow greater separation between seed and shadow, the algorithm Sequential Forward Selection (SFS) was used [9]. This algorithm gives a ranking of the features that most contribute to the separation of classes, in this case, seed and shadow.

After apply SFS algorithm, invariant models obtain a better separability between shadow and seed pixels, however the performance obtained was very low. This means that is not enough the use of invariant color models for shadow pixel segmentation. Because of this, texture features are included in the analysis, and taking Haralick texture descriptors [10] was adopted. These descriptors were selected after comparing their performance with respect to other texture features through SFS algorithm.

The procedure for determining the relevant features is as follows: thousand samples of both shadow and seed were taken in different images (500 of seeds and 500 of shadow), each sample corresponding to a window of 41×41 pixels. 28 texture descriptors were initially computed and by using the algorithm SFS it was established that 9 characteristics allow a good separation of the classes.

2.2 Segmentation through a Neural Classifier

A Multilayer Perceptron was considered as a classifier because this architecture corresponds to a universal function estimator, and permits to classify nonlinearly separable patterns. For the training, the Bayesian Regularization method was adopted [11]. The advantage of this method is that it provides a criterion for determining the number of neurons in the hidden layer based on the effective parameters of the network. The method of obtaining the amount of the hidden layer neurons is the following: Gradually increase the number of the neurons in the hidden layer until the stabilization of the effective parameters of the neural network. The lowest number of neurons in the hidden layer which produces the stabilization of the effective parameters is chosen. Obtaining a reduced number of neurons allows to avoid overfitting and improve network generalization.

The neural classifier has 9 inputs (one for each texture descriptor), 1 output to discriminate between 2 classes (seed and no seed). To segment the seed by the neural network, a grid with cells of 41×41 pixels is set in the image. For each cell, the selected 9 Haralick texture descriptors are computed, which are assigned as inputs to the neural network. The segmentation process is shown in Figure 1. Figure 1(a) shows an example of sed image, and it can be easily seen that there is very little difference between the edge of the seed and the shadow. Figure 1(b) shows the window used in the original image, showing that the size of the window allows a good approximation to the ideal segmentation. Figure 1(c) shows the binary image product of the segmentation, and Figure 1(d) shows

(a) (b) (c) (d)

Fig. 1. Neural Network Segmentation

the edges of the segmented zone in the original image. It is noted that a large amount of shadow has been eliminated, but the method introduces errors in the border areas of the seed and the shadow.

2.3 Segmentation Based on an Invariant Color Model

Invariant color model *c1c2c3* proposed in [4] was adopted due to its good performance against shadows and the highlights elimination. This type of models have this name because is possible obtain the same configuration of color with difference condition in the image such as visualization position and lighting changes. The terms of the model are:

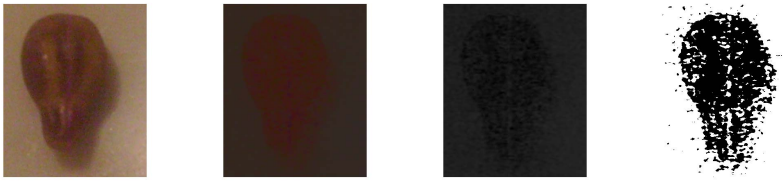
$$c1_{i,j} = \arctan \frac{R_{i,j}}{\max\left(G_{i,j}, B_{i,j}\right)} \tag{1}$$

$$c2_{i,j} = \arctan \frac{G_{i,j}}{\max\left(R_{i,j}, B_{i,j}\right)} \tag{2}$$

$$c3_{i,j} = \arctan \frac{B_{i,j}}{\max\left(G_{i,j}, R_{i,j}\right)} \tag{3}$$

where $R_{i,j}$, $G_{i,j}$ and $B_{i,j}$ represents a pixel in the components red, green and blue in the image I. Applying the above equation over I, a new image I' with three channels(*c1*,*c2* and *c3*) is obtained.

The results of applying this model to an image are presented in figure 2. Figure 2(a) shows the original image of the seed in the RGB model. Figure 2(b) shows the image generated by applying the equations described above. Figure 2(c) shows channel c3 of the invariant color model (this channel is chosen because it had the best ranking when applying the algorithm SFS). Finally, figure 2(d) shows the automatic segmentation of channel c3 using the well-known Otsu method. In the last image, it can be seen that the segmentation of the invariant does not completely eliminate the shadow, and it fails in the inner part of the seed.



(a) (b) (c) (d)

Fig. 2. Invariant Color Model Segmentation

2.4 Segmentation Based on a Hybrid Model

In order to improve the results taken from both techniques separately, it is proposed to combine the supervised and unsupervised segmentation. For the particular case of the models involved in this work, the following combination scheme is proposed:

$$I_{fusion}(i,j) = \begin{cases} 1 \text{ if } I_{seg-inv}(i,j) = 1 \text{ y } I_{seg-net}(i,j) = 1 \\ 0 \quad otherwise \end{cases} \tag{4}$$

where I_{fusion} is the image resulting from the combination of both methods, $I_{seg-inv}$ is the image generated by segmenting in channel $c3$, and $I_{seg-net}$ is the segmentation obtained by the neural network. In order to improve the results obtained to this point, a sequence of binary morphological operations is proposed to eliminate the noise in the image.

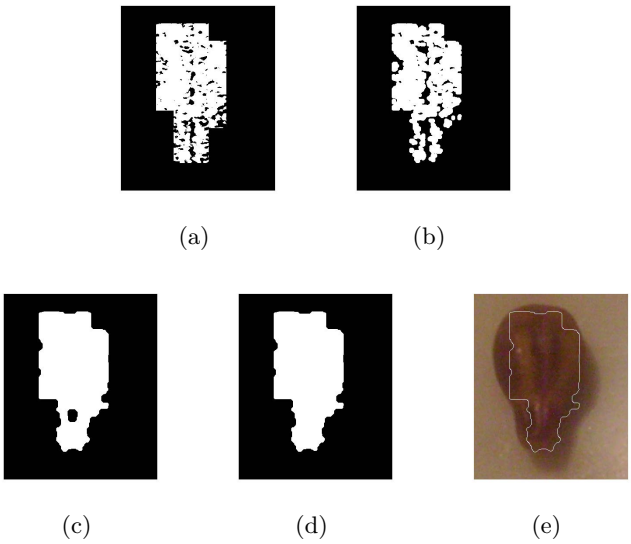
Figure 3 shows the stages of the image enhancement resulting from the combination of methods. Figure 3(a) shows the image fusion, figure 3(b) the morphological opening, figure 3(c) the morphological closing, figure 3(d) the filling of hollows, and finally figure 3(e) the edge of the segmented zone in the original image. It is seen in the last image, that thanks to the improvement, the result is closer to the ideal segmentation than when using both methods separately.

3 Results

To quantify the results obtained through the hybrid method, the ground truth image was obtained manually for a number of seed images (in this case 20 images) as shown in figure 4.

To evaluate the outcome of the methods with respect to the ideal segmentation (ground truth) a performance function widely used in the literature is adopted [12]. This function involves the hits, the false positives and the false negatives. The expression of the performance function is:

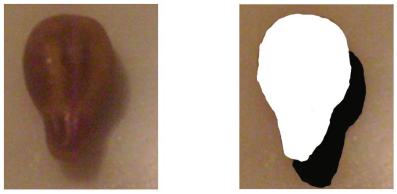
$$\rho = \frac{card(TP)}{card(TP) + card(FP) + card(FN)} \tag{5}$$



(a) (b)

(c) (d) (e)

Fig. 3. Enhancement of the Combination Method



(a) (b)

Fig. 4. Manual Segmentation: (a) Original Image (b) Ground Truth

where ρ is the performance function, card (X) is the cardinal of a set X, TP is the set of true positives, FP is the set of false positives, and FN is the set of false negatives. The performance function ρ is close to 1 if there are many hits and few errors, and is close to 0 if there are few hits and many errors.

Figure 5 shows the performance function for all 20 images showing for each one of them the value of the function for the three methods. Considering the data of an image, the first bar corresponds to the hybrid method, the second bar to the neural network method, the third bar to the invariant. The average of the performance function for the hybrid method is 0.8, for the invariant is 0.75, and 0.7 for the neural network. The hybrid method is superior in the 75% of cases, the invariant in the 25%, and the network in a 0%. The above information numerically shows the benefit that the combination of methods has for the problem.

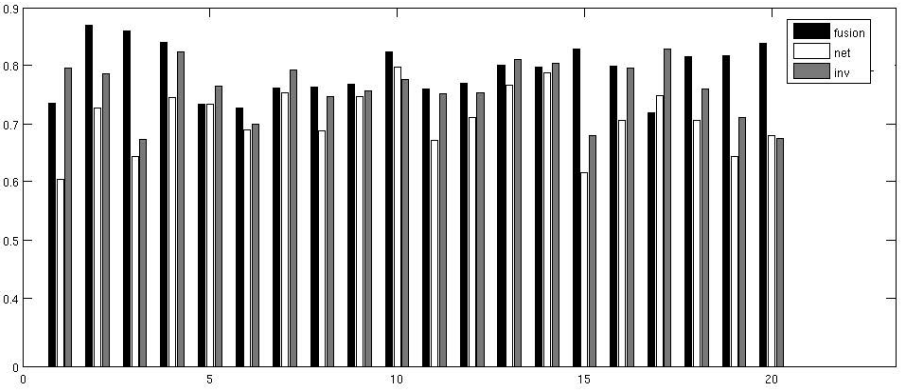


Fig. 5. Evaluation of the Segmentation

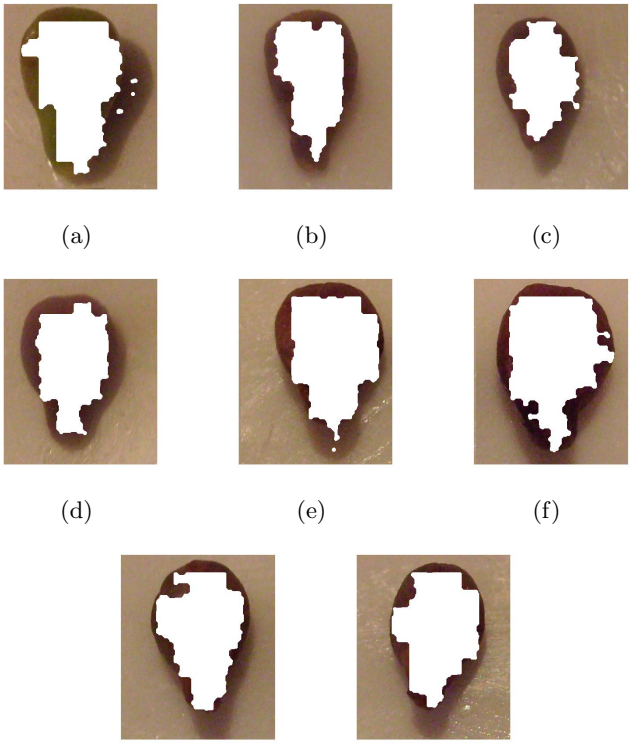


Fig. 6. Global Results

Figure 6 shows the overall results of the method for other images, showing that the segmented zone corresponds to regions belonging to the seed. It is noted that, despite the fact that the edge shape of the segmented region does not coincide perfectly with the actual shape of the seed, the encountered zone is representative of the target region. From the encountered region, color and texture descriptors that may be related to the state of ripeness of the grape can be extracted properly.

4 Conclusions

This paper has proposed a hybrid method to segment shadows in complex images, and its application to the segmentation of wine grape seeds. The method consists of combining the results of an unsupervised segmentation based on invariant models to lightning, and the results of a supervised segmentation based on neural networks.

It is shown that the combination of both methods permits to obtain superior results than when applying the methods independently. The hybrid approach yields appropriate results for the segmentation of the seed, being possible to use the segmented zone in subsequent stages of analysis.

The results of this research are relevant in the sense that they provide the basis for future work in developing a method of classification of seeds based on their ripeness, and in estimating by computational methods and digital images the optimal point of wine grape harvest.

References

1. Ristic, R., Iland, P.: Relationships between Seed and Berries Development of Vitis Vinifera L. cv Shiraz: Developmental Changes in Seed Morphology and Phenolic Composition. Australian Journal of Grape and Wine Reseach 11, 43–58 (2005)
2. Fredes, C., Bennewitz, E.V., Holzapfel, E., Saavedra, F.: Relation between Seed Appearance and Phenolic Maturity: A Case Study Using Grapes cv. Carmenere. Chilean Journal of Agricultural Research 70, 381–389 (2010)
3. Xu, L., Qi, F., Jiang, R., Hao, Y., Wu, G.: Shadow Detection and Removal in Real Images: A Survey (2006)
4. Salvador, E., Cavallaro, A., Ebrahimi, T.: Shadow Identification and Classification using Invariant Color Models. In: Proceedings of IEEE International Conference the on Acoustics, Speech, and Signal Processing, vol. 03, pp. 1545–1548 (2001)
5. Salvador, E., Andrea, A.C., Ebrahimi, T.: Cast Shadow Segmentation using Invariant Color Features. Computer Vision and Image Understanding 95, 238–259 (2004)
6. Finlayson, G.D., Hordley, S.D., Lu, C., Drew, M.S.: On the Removal of Shadows from Images. IEEE Transactions on Pattern Analysis and Machine Intelligence 28, 59–68 (2006)
7. Xu, L., Qi, F., Jiang, R.: Shadow Removal from a Single Image. In: Proceedings of International Conference on Intelligent Systems Design and Applications, vol. 2, pp. 1049–1054 (2006)

8. Gevers, T.: Color-based Object Recognition. Pattern Recognition 32, 453–464 (1999)
9. Jain, A., Robert, P., Duin, M.J.: Statistical Pattern Recognition: A Review. Intelligent Data Analysis 22, 4–34 (1999)
10. Haralick, R.: Statistical and structural approaches to texture. Proceedings of the IEEE (67)
11. Forensee, F., Hagan, M.: Gauss-Newton Approximation to Bayesian Learning. In: Proceedings of the International Joint Conference on Neuronal Networks, vol. 3, pp. 1930–1935 (1997)
12. Grigorescu, C., Petkov, N., Westenberg, M.: Contour Detection based on Nonclassical Receptive Field Inhibition. IEEE Transactions on Image Processing 12, 729–739 (2003)

Author Index