International Conference on Swarm Intelligence Based Optimization

Theoretical advances and real world applications

June 13–14, 2016, UHA, Mulhouse, France

Proceedings of the International Conference on Swarm Intelligence Based Optimization (ICSIBO'2016)

June 13-14, 2016
Mulhouse, France

Sponsored by:

LMIA UNIVERSITÉ HAUTE-ALSACE
LABORATOIRE DE MATHÉMATIQUES, INFORMATIQUE ET APPLICATIONS

FST Faculté des Sciences et Techniques UNIVERSITÉ HAUTE-ALSACE

IUT Institut Universitaire de Technologie de Mulhouse

Foreword

These proceedings include the papers presented at the International Conference on Swarm Intelligence Based Optimization, ICSIBO'2016, held in Mulhouse (France). ICSIBO'2016 is a continuation of the conferences OEP'2003 (Paris), OEP'2007 (Paris), ICSI'2011 (Cergy-Pontoise) and ICSIBO'2014 (Mulhouse).

The aim of ICSIBO'2016 is to highlight the theoretical progress of swarm intelligence metaheuristics and their applications. Swarm intelligence is a computational intelligence technique involving the study of collective behavior in decentralized systems. Such systems are made up of a population of simple individuals interacting locally with one another and with their environment. Although there is generally no centralized control on the behavior of individuals, local interactions among individuals often cause a global pattern to emerge. Examples of such systems can be found in nature, including ant colonies, animal herding, bacteria foraging, bee swarms, and many more. However, swarm intelligence computation and algorithms are not necessarily nature-inspired.

Authors had been invited to present original work relevant to Swarm Intelligence, including, but not limited to: theoretical advances of swarm intelligence metaheuristics ; combinatorial, discrete, binary, constrained, multi-objective, multi-modal, dynamic, noisy, and large-scale optimization ; artificial immune systems, particle swarms, ant colony, bacterial foraging, artificial bees, fireflies algorithm ; hybridization of algorithms ; parallel/distributed computing, machine learning, data mining, data clustering, decision making and multi-agent systems based on swarm intelligence principles ; adaptation and applications of swarm intelligence principles to real world problems in various domains.

Each submitted paper has been reviewed by three members of the international Program Committee. A selection of the best papers presented at the conference and further revised will be published as a volume of Springer's LNCS series.

We would like to express our sincere gratitude to our invited speakers: Brigitte Wolf and Maurice Clerc. The success of the conference resulted from the input of many people to whom we would like to express our appreciation: the members of Program Committee and the secondary reviewers for their careful reviews that ensure the quality of the selected papers and of the conference. We take this opportunity to thank the different partners whose financial and material support contributed to the organization of the conference: Université de Haute Alsace, Faculté des Sciences et Techniques et Institut Universitaire de Technologie de Mulhouse. Last but not least, we thank all the authors who have submitted their research papers to the conference, and the authors of accepted papers who attended the conference to present their work. Thank you all.

June 2016

P. Siarry, L. Idoumghar and J. Lepagnot
Organizing Committee Chairs of ICSIBO'2016

Organization

Organizing Committee Chairs:	P. Siarry, L. Idoumghar and J. Lepagnot
Program Chair:	M. Clerc
Website/Proceedings/Administration:	MAGE Team, LMIA Laboratory

Program Committee

Omar Abdelkafi	Université de Haute-Alsace, France
Ajith Abraham	Norwegian University of Science and Technology, Norway
Antônio Pádua Braga	Federal University of Minas Gerais, Brazil
Mathieu Brévilliers	Université de Haute-Alsace, France
Bülent Catay	Sabanci University, Istanbul, Turkey
Amitava Chatterjee	University of Jadavpur, Kolkata, India
Rachid Chelouah	EISTI, Cergy-Pontoise, France
Raymond Chiong	University of Newcastle, Australia
Maurice Clerc	Independant Consultant, France
Carlos A. Coello Coello	CINVESTAV-IPN, Depto. de Computacion México
Jean-Charles Créput	University of Technologie Belfort-Montbéliard, France
Rachid Ellaia	Mohammadia School of Engineering, Morocco
Frederic Guinand	Université du Havre, France
Jin-Kao Hao	Université d'Angers, France
Vincent Hilaire	Université de Technologie de Belfort-Montbéliard, France
Lhassane Idoumghar	Université de Haute-Alsace, France
Imed Kacem	Université de Lorraine, France
Jim Kennedy	Bureau of Labor Statistics, Washington, USA
Peter Korosec	University of Primorska, Koper, Slovenia
Abderafiaâ Koukam	University of Technologie Belfort-Montbéliard, France
Nurul M. Abdul Latiff	Universiti Teknologi, Johor, Malaysia
Fabrice Lauri	Université de Technologie de Belfort-Montbéliard, France
Stephane Le Menec	RGNC at EADS / MBDA, France
Julien Lepagnot	Université de Haute-Alsace, France
Evelyne Lutton	INRA-AgroParisTech UMR GMPA, France
Vladimiro Miranda	University of Porto, Portugal
Nicolas Monmarché	Université François Rabelais Tours, France
René Natowicz	ESIEE, France
Ammar Oulamara	Université de Lorraine, France
Yifei Pu	Sichuan University, China
Maher Rebai	Université de Haute-Alsace, France
Said Salhi	University of Kent, UK
René Schott	University of Lorraine, France
Patrick Siarry	Université de Paris-Est Créteil, France
Ponnuthurai N. Suganthan	Science and Technology University, Singapore
Eric Taillard	University of Applied Sciences of Western Switzerland
El Ghazali Talbi	Polytech'Lille, Université de Lille 1, France
Antonios Tsourdos	Defence Academy of the United Kingtom, UK
Mohamed Wakrim	University of Ibou Zohr, Agadir, Morocco
Rolf Wanka	University of Erlangen-Nuremberg, Germany

ICSIBO'2016 Scientific program

Monday, June 13, 2016 – Morning

08:30-09:05 – Welcome

09:05-10:20 – Plenary 1
Chair: Brigitte Wolf

"Total Memory Optimiser: A Proof of Concept"
Presented by **Maurice CLERC**

10:20-10:50 – Coffee break

10:50-12:20 – Session 1: Particle Swarm Optimization
Chair: Patrick Siarry

Paper 10: Benoît Beroule, Olivier Grunder, Oussama Barakat, Olivier Aujoulat and Helene Lustig. Particle Swarm Optimization for Operating Theater Scheduling

Paper 11: Rita De Cassia Costa Dias, Hacène Ouzia and Ralf Schledjewskl. Optimization of die-temperature in pultrusion of thermosetting composites for improved cure

Paper 15: Yongqing Zhang, Puyi Fei and Jiliu Zhou. Inference of Large-Scale Gene Regulatory networks using Improved Particle Swarm Optimization

12:20-13:50 – Lunch break

Monday, June 13, 2016 – Afternoon

13:50-16:50 – Social event

Visit of the famous national automobile museum
"Cité de l'Automobile"
Built around the **Schlumpf Collection** of classic automobiles

16:50-17:50 – Session 2: Distributed Algorithms
Chair: Mathieu Brévilliers

Paper 4: Omar Abdelkafi, Lhassane Idoumghar, Julien Lepagnot and Mathieu Brévilliers. Data exchange topologies for the DISCO-HITS algorithm to solve the QAP

Paper 9: Hongjian Wang, Abdelkhalek Mansouri, Jean-Charles Créput and Yassine Ruichek. Distributed Local Search for Elastic Image Matching

17:50-18:20 – Coffee break

18:20-19:20 – Session 3: Parallel Algorithms
Chair: Julien Lepagnot

Paper 13: Mathieu Brevilliers, Omar Abdelkafi, Julien Lepagnot and Lhassane Idoumghar. Fast Hybrid BSA-DE-SA Algorithm on GPU

Paper 19: Dahmri Oualid and Baba-Ali Ahmed Riadh. A New Parallel Memetic Algorithm to Knowledge Discovery in Data Mining

20:30-22:30 – Gala dinner at "Chez Henriette"

Tuesday, June 14, 2016 – Morning

08:00

09:00

09:05-10:20 – Plenary 2
Chair: Maurice Clerc

"Inspiration by Swarms"
Presented by **Brigitte WOLF**

10:00

10:20-10:50 – Coffee break

10:50-12:20 – Session 4: Applications
Chair: Lhassane Idoumghar

11:00

Paper 7: Charaf Eddine Khamoudj, Karima Benatchba and Tahar Kechadi. Classical Mechanics Optimization for image segmentation

Paper 14: Halil Alper Tokel, Gholamreza Alirezaei and Rudolf Mathar. Modern Heuristical Optimization Techniques for Power System State Estimation

Paper 17: Youcef Abdelsadek, Kamel Chelghoum, Francine Herrmann, Imed Kacem and Benoît Otjacques. On the community identification in weighted time-varying networks

12:00

12:20-13:50 – Lunch and conference end

13:00

Guest speakers

Maurice CLERC

Maurice CLERC was working with France Telecom R&D as research engineer (optimization of telecommunications networks). In 2005 he has been awarded with James Kennedy by IEEE Transactions on Evolutionary Computation for their 2002 paper on Particle Swarm Optimization (PSO). He is now retired but still active in this field: a book about PSO in 2005 (translated into English in 2006), a book in 2015 about guided randomness in optimization (translated into English), several papers in international journals and conference proceedings, external examiner for PhD theses, reviewer and member of editorial board and program committee for conferences and journals (IEEE TEC Best Reviewer Award 2007), co-webmaster of the Particle Swarm Central.

Abstract of the plenary talk entitled "Total Memory Optimiser: A Proof of Concept"

For most usual optimisation problems, the Nearer is Better assumption is true (in probability), This property is taken into account by the classical iterative algorithms, either explicitly or implicitly, by forgetting some information collected during the process, assuming it is not useful any more. However, when the property is not globally true, i.e. for deceptive problems, it may be necessary to keep all the sampled points and their values, and to exploit this increasing amount of information. Such a basic Total Memory Optimiser is presented. We show on an example that it can outperform classical methods on deceptive problems. As it is very computing time consuming as soon as the dimension of the problem increases, a few compromises are suggested to speed it up.

Brigitte WOLF

After studying industrial Design and Psychology, Brigitte Wolf has had a varied international career as project manager, consultant, researcher and lecturer. In 1991 she was awarded the first professorship for design management in Germany at the University of Applied Sciences Cologne. Since october 2006 Brigitte Wolf has led the Centre for Applied research in Brand, reputation and Design management (CBrD) at iNHollAND University of Applied Sciences in Rotterdam. In 2007 she became professor of design theory at the University of Wuppertal, with a focus on the planning, methodology and strategy of design management.

Abstract of the plenary talk entitled "Inspiration by Swarms"

The hypothesis of the lecture is, that swarm intelligence will enable companies to operate successful in the future by integrating design strategy into their business strategy. Characteristics of swarm behavior and characteristics of human behavior will be discussed to find out, how principles of swarm behavior can be used to improve design strategies in corporate businesses. Some examples that adapted principles of swarm intelligence will be presented. Finally an example of the swarm inspired strategic approach for a company we will work with in the winter term will be given.

Accepted papers and abstracts

Table of Contents

Particle Swarm Optimization for Operating Theater Scheduling

Benoit Beroule[1], Olivier Grunder[1], Oussama Barakat[2], Olivier Aujoulat[3], and Helene Lustig[3]

[1] Univ. Bourgogne Franche Comté , UTBM, IRTES-SET, 90010 Belfort, France.
{benoit.beroule,olivier.grunder}@utbm.fr
http://www.utbm.fr
[2] Nanomedecine Lab, University of Franche Comté , 25000 Besançon, France.
oussama.barakat@univ-fcomte.fr
http://www.univ-fcomte.fr
[3] GHRMSA, Mulhouse hospital center 68000 Mulhouse, France.
{aujoulato,lustigh}@ch-mulhouse.fr
http://www.ch-mulhouse.fr

Abstract. The hospital surgical procedures scheduling problem is a well-known operational research issue. In this paper, we propose a particle swarm optimization (PSO) based algorithm to solve this problem for the purpose of reducing surgical devices utilization and thus improve the sterilization service efficiency in a hospital context. we define a computation space to simplify calculation steps. Moreover, we detail the modeling and provide a study on the PSO factors and their impact on the final results then finally determine the best value for each factor to solve this particular problem.

Keywords: optimization; health care ; particle swarm optimization; operating theater scheduling

1 Introduction

The constant progresses made in the health care sector keep improving people's life expectancy. In the other hand, the average time spent in hospital centers for a person is inexorably rising. To be able to meet this increasing demand, the hospital sector looks towards the operational research sector. Actually, numerous hospital aspects could be improved by using appropriate management methods such as nurses assignment [5], materials transportation, patients routing and much more. This paper focuses on the surgical procedures scheduling problem which is a major issue of hospital management and a widely studied problem [11]. Eight main performance criteria are commonly used in the literature to evaluate operating room scheduling procedures [2] : waiting time, throughput, utilization, leveling, makespan, patient deferrals, financial measures and preferences. A method was developed to maximize operating room utilization considering allocating block time and thus, correctly manage elective (non urgent) patients [6].

The non-elective surgery must also be taken into account, this is why a stochastic dynamic programming model was implemented to schedule elective surgery under uncertain demand for emergency surgery [7]. Moreover, some industrial management methods may be adapted to the hospital sector. The scheduling problem may be identified to a hybrid flow shop to determine a $o(n^2)$ complexity dedicated heuristic [12]. When applying to an important hospital center, exact methods may require prohibitive computation times. Therefore, some studies deal with approximate methods as a tabu search to establish a surgical procedures schedule according to different planning policies [8]. It is against this background that we propose in this paper, a particle swarm optimization based scheduling method. The particle swarm optimization (PSO) is a parallel evolutionary computation meta heuristics invented by Kennedy and Eberhart [10, 13, 9] which is based on insects social behavior. Particles are created in the solutions space and share information to move and converge towards best solutions. Numerous papers deal with PSO improvements or practical applications. A PSO parameters choice method was defined to improve convergence rate and discuss on each parameter utility [3]. Indeed, the parameters greatly affect the solutions consistency. Consequently, some papers studied their impact in a mathematical [15] or empirical [14] way. In this paper, we propose a detailed PSO method to solve the operating block scheduling problem taking into account medical devices utilization as well as an empirical selection and a discussion on the parameters.

2 Studied problem

When considering the operating theater scheduling problem, numerous aspects of the hospital sector may be taken into account (nurses availability, patient types, material flows...). In this study, we focus on the medical devices utilization cycle. Medical devices are packaged into "boxes" which are opened and prepared by a nurse before each surgical procedure. After being used, the devices are predesinfected in a dedicated place of the operating theater before being repackaged in their respective box then resent to the sterilization service which is commonly a part of the hospital pharmacy. The sterilization service receive the boxes and perform several operations. First, the material is separately cleaned thanks to washing machines. Then, the human agents repack the medical devices into the boxes according to a precise protocol depending on the surgical operation type. Finally, the boxes are sterilized thanks to autoclaves and resent to the operating theater when their temperature drops enough (or stored in the service if they are not immediately needed).
By working on the surgical procedures scheduling, we hope to improve two distinct aspects of the sterilization service. In one hand, the quantity of needed boxes could be reduced which implies a better reaction when facing an urgency case. In the other hand, the working activity of the sterilization service may be more heterogeneously distributed to avoid any burst in activity.

3 Particle Swarm Optimization modeling

In this section, we present a PSO based algorithm the purpose of which is to solve the surgical procedures scheduling problem by minimizing surgical devices utilization. To be efficient, this algorithm must provide solutions as near as possible to those provided by the MILP model [1].

3.1 Modeling

Implementing a PSO algorithm implies determining the modeling of the particles which will explore the solutions space. Our purpose is to determine a one week surgical procedures planning by determining the starting date of each operation. Furthermore, the duration time of a procedure is not a decision variable and may mainly depends on the patient physical characteristics, the pathology type or the surgeon habits. In these conditions, the starting dates are sufficient to establish a complete planning with approximate duration times.
We first define the modeling parameters divided into two sections: MILP relative parameters and PSO relative parameters (some of them will be detailed afterward).
MILP relative parameters:

- n : The amount of surgical procedures waiting to be scheduled.
- di : The starting date of the surgical procedure i ($1 \leq i \leq n$).
- T^o : The operating theater opening date ($0 \leq T^o \leq T^c$).
- T^c : The operating theater closing date ($T^o \leq T^c \leq T$).
- $T = 24h$: The duration of a day.

PSO relative parameters:

- m : The amount of particles generated for the PSO algorithm.
- p : The amount of steps performed by the PSO algorithm.
- X_j^k : The position vector of the particle j at step k.
- V_j^k : The velocity vector of the particle j at step k.
- L_j : The best solution founded by the particle j.
- G : The best solution founded by the particles.
- ω : The inertia factor.
- ϕ_1 : The personal memory factor.
- ϕ_2 : The common knowledge factor.
- S_1 : The solutions space.
- S_2 : The computation space.
- r_1^k, r_2^k : Vectors of random generated float from 0 to 1.

Hence, each particle is represented by its position and velocity. The position is a n-tuple as shown in equation (1).

$$X_j^k = (d_1, d_2, ..., d_n) \tag{1}$$

With this modeling, the particles progress in a n dimensional space. A movement along a dimension i represents a modification of the corresponding starting date

d_i. To initialize the PSO, m particles will be generated with random starting dates distributed during the concerning week and random initial velocities V_j^0. m must be big enough to create a set of particles covering the entire solution space. At each step k, a particle represents a particular solution according to its position in the solution space.
During each step of the PSO algorithm, the particles will communicate to share information and update their own positions according to their own knowledges and the common knowledge of the best solution. The details of the new position computation is given in equation (2) and (3) [10].

$$V_j^{k+1} = \omega V_j^k + \phi_1 r_1^k (L_j - X_j^k) + \phi_2 r_2^k (G - X_j^k) \tag{2}$$

$$X_j^{k+1} = X_j^k + V_j^{k+1} \tag{3}$$

L_j and G represent the position vectors of the best solutions founded by the particle j and by the entire set of particles respectively. They are updated at each step if needed. ω represents the system global inertia. A high inertia value implies a better solution space exploration at the expense of the convergence speed. ϕ_1 and ϕ_2 represent the personal memory factor and the common knowledge factor respectively. If ϕ_1 is set to a high value, each particle will be more attracted by its own best already visited position L_j. If ϕ_2 is set to a high value, each particle will be more attracted by the best already visited position among every visited positions of every particles G.
After p steps, the solution corresponding to the best visited position among every particles is considered as the PSO algorithm output . p must be big enough to allow the particles to converge toward one or several extrema, but not too big to prevent the machine from prohibitive computation time.

3.2 Computation space

Among other factors, the PSO efficiency depends on the solution space topology and the fitness function behavior. Here we define the solution space S_1 as all possible dates combinations in a week (equation (4)).

$$S_1 = \{(d_1, d_2, ..., d_n) | \forall i \in [\![1, n]\!], 0 \leq d_i \leq 5 \times T \ \ , \ \ T_o \leq d_i \mathbf{mod}(T) < T_c\} \tag{4}$$

In this scheduling problem, the fitness function evaluates the number of needed boxes to respect a given schedule. The problem is that S_1 is a discrete subset of $\mathbb{R}^n$, this topology particularity prevents the particles from moving in a continuous way. To improve the PSO efficiency, we consider a new space, S_2 (continuous subset of $\mathbb{R}^n$), which will be called the "computational space" (equation (5)).

$$S_2 = \{(d_1, d_2, ..., d_n) | \forall i \in [\![1, n]\!], 0 \leq d_i < 5 \times (T_c - T_o)\} \tag{5}$$

S_2 and S_1 are homeomorphic, therefore there is a bijective continuous function (equations 6 and 7) to translate the straight forward readable solution from S_1 to

S_2 where the computation is easier. When the computation is over, the solutions may be translated back from S_2 to S_1 (equations 8 and 9).

$$\begin{aligned} f: \quad & S_1 \to S_2 \\ & (d_1, d_2, ..., d_n) \mapsto f((d_1, d_2, ..., d_n)) = (d'_1, d'_2, ..., d'_n) \end{aligned} \tag{6}$$

$$d'_i = (d_i - T^o) - \frac{d_i}{T} \times (T + T^o - T^c) \qquad (\frac{a}{b} \textit{ is the euclidian division}) \tag{7}$$

$$\begin{aligned} f^{-1}: \quad & S_2 \to S_1 \\ & (d'_1, d'_2, ..., d'_n) \mapsto f^{-1}((d'_1, d'_2, ..., d'_n)) = (d_1, d_2, ..., d_n) \end{aligned} \tag{8}$$

$$d_i = (d'_i + T^o) + (T + T^o - Tc) \times (d'_i \mathbf{mod} [T^c - To]) \tag{9}$$

4 Experimentation

To ensure the reliability of the results obtained by the PSO, each parameter must be calibrated according to the current scheduling problem. Hence the purpose of this section is to determine each parameter best value to improve the PSO algorithm error ratio.

4.1 Determining best parameters

In order to improve the PSO efficiency, we study the impact of the ω, ϕ_1 and ϕ_2 factors on the solution provided by the PSO algorithm. Therefore, we implement a parameters evaluation algorithm (Fig 1). After performing this algorithm for a scenario S, we obtain a 3-dimensional data structure F_s containing an average on *NbIter* iterations of the best solutions fitness obtained by the PSO for any triplet $(\omega, \phi_1, \phi_2) \in P$ (define in equation (10)).

$$P = P_\omega \times P_{\phi_1} \times P_{\phi_2} \tag{10}$$

$$P_\omega = \{\omega \in \mathbb{R} | \exists i \in \mathbb{N}, \omega = \omega_{start} + i \times \omega_{step} \ , \ \omega \leq \omega_{end}\} \tag{11}$$

$$P_{\phi_1} = \{\phi_1 \in \mathbb{R} | \exists i \in \mathbb{N}, \phi_1 = \phi_{1\,start} + i \times \phi_{1\,step} \ , \ \phi_1 \leq \phi_{1\,end}\} \tag{12}$$

$$P_{\phi_2} = \{\phi_2 \in \mathbb{R} | \exists i \in \mathbb{N}, \phi_2 = \phi_{2\,start} + i \times \phi_{2\,step} \ , \ \phi_2 \leq \phi_{2\,end}\} \tag{13}$$

Therefore, we define a set of representative scenarios $S = \{s_1, s_2, ..., s_l\}$, and obtain the best parameters according to equation (14).

$$(\omega_{best}, \phi_{1\,best}, \phi_{2\,best}) = arg\,min(\sum_{s \in S} F_s(i, j, k)) \tag{14}$$

Here we define the ranges of value for each parameter with:
$\omega_{sart} = \phi_{1\,start} = \phi_{2\,start} = 0.2$,
$\omega_{step} = \phi_{1\,step} = \phi_{2\,step} = 0.2$,
$\omega_{end} = \phi_{1\,end} = \phi_{2\,end} = 2.0$,
to obtain equation (15).

$$(\omega_{best}, \phi_{1\,best}, \phi_{2\,best}) = (0.2, 1.2, 1.0) \tag{15}$$

Fig. 1. PSO best parameters evaluation algorithm

```
const
     NbIter: Integer; S: Scenario;
     omegaStart, omegaStep, omegaEnd: Real
     phi1Start, phi1Step, phi1End: Real
     phi2Start, phi2Step, phi2End: Real
var
     i := omegaStart; j := phi1Start; k := phi2Start;
     it: integer;
     Fs: Real 3 dimensional data structure;
begin
     repeat
          repeat
               repeat
                    it := 1;
                    Fs(i,j,k) := 0;
                    repeat
                         Fs(i,j,k) := Fs(i,j,k) + PSOBestSolutionFitness(i,j,k,S);
                         it := it + 1;
                    until it > NbIter
                    Fs(i,j,k) := F(i,j,k) / NbIter;
                    k := k + phi2Step
               until k > phi2End
               j := j + phi1Step
          until j > phi1End
          i := i + omegaStep;
     until i > omegaEnd
end
```

We do not assure that the previously determined parameters are the best choice to converge toward the best solution but we assume they are an interesting alternative considering the fact that only 2 hours (with $NbIter = 50$) was needed to compute them. Let us consider the consistency of our results. A theoretical approach leads to define the PSO factors by the equations $\phi_1 = \phi_2 = \phi$ and $\phi = \omega \times (2/0.97725)$ or $\phi \approx 2 \times \omega$ [4], this is why we first decoded to use the parameters $(\omega, \phi_1, \phi_2) = (1.0, 2.0, 2.0)$. From the empirical results of testing, two observations can be made. First $\phi_{1\,best} \approx \phi_{2\,best}$ (indeed $\phi_{1\,best} = 1.0$ and $\phi_{2\,best} = 1.2$). However the inertia factor $\omega_{best} = 0.2$ is smaller than the expected value (about 1.0). To understand this result, let us remind the impact of this parameter on the global system. The inertia factor represents the particles capacity of "quickly" change their directions, therefore, the bigger inertia factor, the more the solution space is explored (but the convergence rate may decreased). Nevertheless, the solution space of the current problem contains several non-neighboring optimal solutions (for instance, inverting two surgical procedures of same duration provide an other solution with identical fitness). Consequently, the exploration of the entire solution space is not crucial, hence the inertia factor does not need to be set to a high value in this context.

4.2 Results

Table 1. Number of boxes needed to respect each scenario depending parameters value and MILP model

scenario	procedures	PSO_1	PSO_2	MILP	scenario	procedures	PSO_1	PSO_2	MILP
1	6	2.00	2.00	2	17	22	5.01	5.00	-
2	7	2.00	2.00	2	18	23	5.44	5.15	-
3	8	2.00	2.00	2	19	24	5.75	5.63	-
4	9	2.04	2.00	2	20	25	6.00	5.97	-
5	10	2.66	2.37	2	21	26	6.02	6.00	-
6	11	3.00	3.00	3	22	27	6.11	6.05	-
7	12	3.00	3.00	3	23	28	6.66	6.24	-
8	13	3.00	3.00	3	24	29	6.98	6.90	-
9	14	3.21	3.05	3	25	30	7.01	7.00	-
10	15	3.75	3.61	3	26	31	7.16	7.01	-
11	16	4.00	4.00	4	27	32	7.54	7.20	-
12	17	4.00	4.00	4	28	33	7.83	7.73	-
13	18	4.02	4.00	4	29	34	7.96	7.95	-
14	19	4.28	4.11	4	30	35	8.05	7.99	-
15	20	4.98	4.93	4	31	36	8.19	8.01	-
16	21	5.00	5.00	5	32	37	8.62	8.24	-

We evaluate the schedules provided by two different PSO algorithms. PSO_1 uses the classical parameters $(\omega, \phi_1, \phi_2) = (1.0, 2.0, 2.0)$ while PSO_2 uses the parameters $(\omega, \phi_1, \phi_2) = (0.2, 1.2, 1.0)$. The table 4.2 summarizes the performances of each algorithm by displaying the minimum average number of boxes needed to respect the best schedule obtained. We compare it to the exact solution obtained with a MILP model (when the computation time is under 1 hour) on 32 scenarios containing from 6 to 37 surgical procedures.
Note that each instance of scenarios from 1 to 16 (left table) is solved with $n = 100$ particles and $m = 10$ cycles. By increasing the amount of particles or the number of cycles, the solutions quality will be improved but the algorithm could not be easily compared. The scenarios from 17 to 32 (right table) are solved with $n = 1000$ and $m = 100$ for a computation time of few seconds for each of them.

5 conclusion

The PSO based algorithm details in this paper provides interesting results to solve the surgical procedures scheduling problem. It may be used as a replacement for the MILP model when the amount of concerned procedures is to high to be computed in a reasonable amount of time. An improvement of this method might be to applied an effect zone to each particle and then only consider the neighborhood of each of them to compute its next step position. As said before, we are dealing with a multi nodal problem, there is therefore every chance that

using a neighborhood based method allows to determine several best solutions. The next step of the study is now to implement a real time algorithm to update a schedule according to the new prescribe procedures of each day and test it in a real hospital context.

References

1. Benoit Beroule, Olivier Grunder, Oussama Barakat, Olivier Aujoulat, and Helene Lustig. Ordonnancement des interventions chirurgicales dun hopital avec prise en compte de létape de stérilisation dans un contexte multi-sites.
2. Brecht Cardoen, Erik Demeulemeester, and Jeroen Beliën. Operating room planning and scheduling: A literature review. *European Journal of Operational Research*, 201(3):921–932, 2010.
3. Maurice Clerc and James Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.
4. Maurice Clerc and Patrick Siarry. Une nouvelle métaheuristique pour l'optimisation difficile: la méthode des essaims particulaires. *J3eA*, 3:007, 2004.
5. Jérémy Decerle, Olivier Grunder, Amir Hajjam El Hassani, and Oussama Barakat. Optimisation de la planification du personnel dun service de soins infirmiers à domicile.
6. Franklin Dexter, Alex Macario, Rodney D Traub, Margaret Hopwood, and David A Lubarsky. An operating room scheduling strategy to maximize the use of operating room block time: computer simulation of patient scheduling and survey of patients' preferences for surgical waiting time. *Anesthesia & Analgesia*, 89(1):7–20, 1999.
7. Yigal Gerchak, Diwakar Gupta, and Mordechai Henig. Reservation planning for elective surgery under uncertain demand for emergency surgery. *Management Science*, 42(3):321–334, 1996.
8. Arnauld Hanset, Hongying Fei, Olivier Roux, David Duvivier, and Nadine Meskens. Ordonnancement des interventions chirurgicales par une recherche tabou: Exécutions courtes vs longues. *Logistique et Transport LT07*, 2007.
9. James Kenndy and RC Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.
10. James Kennedy. Particle swarm optimization. In *Encyclopedia of machine learning*, pages 760–766. Springer, 2011.
11. Nathalie Klement. *Planification et affectation de ressources dans les réseaux de soin: analogie avec le problème du bin packing, proposition de méthodes approchées*. PhD thesis, Université Blaise Pascal-Clermont-Ferrand II, 2014.
12. NH Saadani, A Guinet, and S Chaabane. Ordonnancement des blocs operatoires. In *MOSIM: Conference francophone de MOdélisation et SIMulation*, volume 6, 2006.
13. Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69–73. IEEE, 1998.
14. Yuhui Shi and Russell C Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary programming VII*, pages 591–600. Springer, 1998.
15. Ioan Cristian Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6):317–325, 2003.

Optimization of die-temperature in pultrusion of thermosetting composites for improved cure

RITA DE CASSIA COSTA DIAS[1] *, HACENE OUZIA[2] and RALF SCHLEDJEWSKI[1]

[1]Chair of Processing of Composites, Department Polymer Engineering and Science,

Montanuniversität Leoben, Otto Glöckel-Straße 2, 8700 Leoben, Austria

[2]Université Pierre et Marie Curie, 4 place Jussieu, 75252 Paris, France

* Corresponding author (RitadeCassia.CostaDias@unileoben.ac.at)

Keywords: Nodal control volume, Pultrusion, Thermal analysis, Degree of cure

Abstract

In this work, we will present a swram optimization based approach to optimize die-temperature and pull-speed in pultrusion of thermosetting composite. Pultrusion is a composite manufacturing technique for processing continuous composite profiles with a constant cross section. The materials which are used for pultrusion in the industry are continuous glass fibers with polyester or epoxy resins. During composite processing, the reinforcing fibers are impregnated with a liquid resin in an injection box or resin bath, fibers and resin are preheated in a mold in which the curing process takes place. High productivity and low operating costs are the main advantages of this processing method. During processing, the heat flux provided by the mold must be sufficient to promote the polymerization reaction of the thermosetting matrix (curing). Furthermore, curing of a composite should be uniform and sufficient in order to provide a good quality of the end product. The exothermic character of the curing reaction induces, inside the composite, exceed temperatures. This temperature rise can cause degradation of the final product. Also, in pultrusion process, transport phenomena are involved and mathematical models are necessary to predict the physico-chemical behavior of the process. For such studies, the region enclosed by the mold is usually considered the main part of the process in which the curing reaction occurs and heat is transfered. Thus, the optimization process is quite important for the prediction of die-heating temperature and pull-speed.

To compute the die-heating temperatures and pull speed that give the best degree of cure of the composite we will use the function, given in [1], relating die-heating temperatures and pull speed to the degree of cure of the composite. A particle swarm based approach (see [2]) will be used to optimize this function. The best die-heating temperatures and pull speed found will be used again (as initial boundary condition) to compute the degree-of-cure profiles in the composite (at the exit section of the mold). This optimization step will

be executed several times until a measure of uniformity attains a certain threshold (the same measure as in [1] will be used).

As computational results, the die-heating environment will be optimized for few cases (different geometries) with different initial temperatures for a glass/epoxy. A general-purpose finite element software, ANSYS-16.2, is used in order to perform three dimensional conductive heat transfer analysis and the MATLAB PSO solver will be used to compute the die-heating temperatures and pull-speed. The solutions obtained using the PSO solver will be compared (when it is possible) to the exact solution of the optimization problem.

References

[1] Li J, Joshi SCJ, Lam YC, Curing optimization for pultruded composite sections. Composites Science and Technology 2002;62: 457-467.

[2] Kennedy, J., Eberhart, R., Particle Swarm Optimization. In: Proc. IEEE International Conference on Neural Networks 1995.

Acknowledgement

Research stay of RITA DE CASSIA COSTA DIAS at the Montanuniversität Leoben is funded by (CAPES) Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brazil

Inference of Large-Scale Gene Regulatory networks using Improved Particle Swarm Optimization

Yongqing Zhang[1, 2], Yifei Pu[1], Jiliu Zhou[3, 1, ‡]

[1] College of Computer Science, Sichuan University, Chengdu, 610065, PR China

[2] Department of Bioengineering, University of California, San Diego, La Jolla, CA 92093, USA

[3] Department of Computer Science, Chengdu University of Information Technology, P.R. China, 610225

‡ Corresponding author: zhoujl@scu.edu.cn

Abstract: Gene regulatory networks provide a systematic view of molecular interactions in a complex system. One of the most challenging problems in systems biology is the process of inferring large-scale gene regulatory networks. Here we adopted a differential equation model to represent gene networks and used Improved Particle Swarm Optimization to infer the appropriate network parameters. Our method attempted to generate a higher diversity of particles during the evaluation. The swarm was first divided into several groups, then each particle learned from other better particles in their current group. Finally, the crossover operator was used to randomly select two particles in the current group. To validate the proposed methods, three low-dimensional tests and three high-dimensional tests have been conducted; the searching dimensionality is 25, 64, 100, and 225, 400, 900 respectively. The results show that the proposed methods can be used to infer differential equation models of gene regulatory networks efficiently and with high stability.

Keywords: large-scale gene regulatory network, particle swarm optimization, time-series.

1 Introduction

Gene expression is the process of generating functional gene products, such as mRNA and protein. The level of gene functionality can be measured from gene expression data produced using microarrays or gene chips[1, 2]. Measuring the levels of gene expression under different conditions is vital for medical diagnosis, treatment, and drug design applications[3]. Many gene expression experiments produce time-series data with only a few time points due to high measurement costs. Therefore, it becomes significant to predict the behavior of gene regulatory networks (GRNs) through modern computing technology. Recently, many algorithms and mathematical models have been proposed to predict gene regulatory networks from time-series data, such as Boolean networks[4], Dynamic Bayesian networks[5], neural networks[6], different equations models[7, 8] and so on. In the above-mentioned GRNs inference, the most important steps are choosing a network model and determining the best parameters of the network model using the gene expression time-series data. Several evolutional algorithms have been proposed to deduce the GRNs[9, 10].

Among the many evolutionary algorithms, Particle swarm optimization (PSO) is one of the most powerfully used swarm intelligence algorithms, originally attributed to Eberhart and Kennedy[11]. The algorithm is based on a simple mechanism that mimics swarm behaviors of social animals, such as bird flocking. The PSO comprises of many particles, and each of the particles has a position. This position can be compared to the particle's best position and the swarm's best position. Each particle also has a velocity, which can adjust the particle's relative

position closer to the best position in the swarm.

Each particle's velocity and position will be changed according to the following equations:

$$V_{i,j}(t+1) = \omega_t \cdot V_{i,j}(t) + c_1 \cdot \varphi_1(t) \cdot \left(Pbest_{i,j}(t) - X_{i,j}(t)\right) + c_2 \cdot \varphi_2(t) \cdot (Gbest(t) - X_{i,j}(t)) \quad (1)$$

$$X_{i,j}(t+1) = X_{i,j}(t) + V_{i,j}(t+1) \quad (2)$$

where t is the iteration number, $V_{i,j}(t)$ and $X_{i,j}(t)$ represent the velocity and position of the ith particle in the jth dimension, respectively. ω_t is termed the inertia weight, c_1 and c_2 are the acceleration coefficients, $\varphi_1(t)$ and $\varphi_2(t)$ are two randomly generated numbers with [0,1], $Pbest_{i,j}(t)$ is the best position for particle i and $Gbest(t)$ is the best position the swarm has obtained.

Due to its conceptual simplicity and high search efficiency, PSO has been widely used in many applications, such as optimization[12, 13], classification[14], complex network clustering[15, 16] and so on. However, it has been found that PSO performs poorly when the optimization problem has a large number of local optima or is high dimensional[17]. Classic PSO will often reach a local minimum as its final solution.

Because of the strong influence of the global best position, $Gbest$, on the convergence speed[18], $Pbest_i$ is very likely to have a value similar to or even the same as $Gbest$, and this will reduce the swarm diversity. In order to increase the diversity of swarm, we propose three aspects to improve PSO in our paper.

1) In each interaction, all particles will divide into several groups after being ordered by fitness. The velocity and position of each particle will be updated in each group, not in the swarm. In this way, we have many small swarms to search the best result.
2) In our paper, the update of velocity does not depend on $Gbest$ and $Pbest_i$. Each particle can choose any better particle as $Gbest$, and choose the average of each group instead of $Pbest_i$.
3) After the above step, two particles are randomly chosen as a pair, and the crossover operator is applied on these two particles with probability $P_{crossover}$ in each group.

2 Materials and methods

2.1 Model

As mentioned earlier, time series data is an important tool to model gene expression. Due to the complexity of GRNs, differential equations are a popular choice to be used in models used to infer dynamic system gene regulation.

The gene regulatory network containing n genes is described by the following discrete time non-linear stochastic dynamical system[19]:

$$x_i(k) = \sum_{j=1}^{n} a_{ij} f_j\left(x_j(k-1)\right), \; i = 1,2,\dots,n, k = 1,2,\dots,m \quad (3)$$

where $x_i(k)$ is the ith actual gene expression level at time k, n is the number of genes and m is the number of measured time points. $A = (a_{ij})_{n\times n}$ represents the non-linear regulatory relationship among genes, and the nonlinear function $f_j(x_j)$ is given by

$$f_j(x_j) = \frac{1}{1+e^{-x_j}} \quad (4)$$

So in our model, A are the parameters to be identified.

2.2 Fitness functions

Since our goal is to find the best parameters A for the GRNs, it is necessary to formulate this as an optimization problem. The fitness function that is used to measure the deviation of the GRNs prediction value from the real measurement is defined as

$$min\ Fitness = \frac{1}{mn}\sum_{k=1}^{m}\sum_{i=1}^{n}(x_{i,pre}(k) - x_{i,real}(k))^2 \quad (5)$$

where $x_{i,pre}(k)$ represents the prediction value of x_i at the time point k and the $x_{i,real}(k)$ represents the real value of x_i at the time point k.

2.3 Improved PSO (IPSO)

2.3.1 The overall framework

Like the PSO algorithm, a swarm $P(t)$ has N particles that represent candidate solutions, where N is the swarm size and t is the generation index. Each particle has a M-dimensional position, $X_i(t) = \left(x_{i,1}(t), x_{i,2}(t) \dots, x_{i,M}(t)\right), i = 1,2 \dots N$, and a M-dimensional velocity vector $V_i(t) = (v_{i,1}(t), v_{i,2}(t) \dots, v_{i,M}(t))$ where M is the number of optimized parameters.

Because of the strong influence of the global best position, $Gbest$, we don't use $Gbest$ to update particles. In each generation, there are three steps to update particles. Firstly, the particles in $P(t)$ are sorted according to an increasing order of the fitness value of the particles. Secondly, all the particles N are mode K to N/K groups. Consequently, we can update particles in N/K groups. In each group, the best particle will be passed directly to the next generation and the other particles will update their position and velocity by learning from a particle which has better fitness and average position values for their current group, mentioned later in Section 2.3.2. Thirdly, all the particles will update their position again when the crossover operator is applied to the current group. The IPSO technique can be described in the following steps in Figure .1.

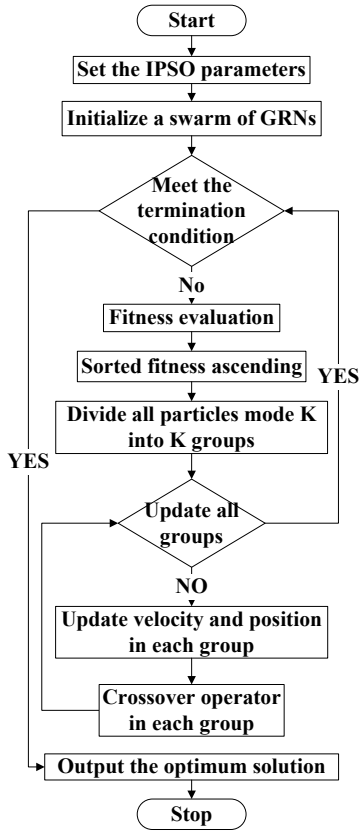


Fig. 1. The flowchart of the IPSO algorithm.

2.3.2 Update of velocity and position

It is known that in a group, a particle trying to learn from any better individuals will also be influenced by other individuals in the current group. So we propose a new learning method; that each particle in a group will learn from better individuals and be influenced by the average position of the current group. Let us denote the velocity and position of the ith particle in the jth dimension in generation t in each group in the following manner:

$$V_{i,j}(t+1) = \omega_t \cdot V_{i,j}(t) + c_1 \cdot \varphi_1(t) \cdot \left(X_{k,j}(t) - X_{i,j}(t)\right)$$

$$+\alpha \cdot c_2 \cdot \varphi_2(t) \cdot (\overline{X}_j(t) - X_{i,j}(t)) \quad (6)$$

$$X_{i,j}(t+1) = X_{i,j}(t) + V_{i,j}(t+1) \quad (7)$$

In the above updating mechanisms, $V_{i,j}(t+1)$ consists of three parts. The first part is the same as Classic PSO, while the other two parts are different. In the second part, instead of learning from personal best, $Pbest$, as done in Classic PSO, the particle i learns from any better particle $X_{k,j}(t)$ in the current group (except the best particle in the group). Therefore, the i satisfies $1 < i \leq N/K$ and k satisfies $1 \leq k < i$. In the third part, since the individual will be influenced by other individuals in a group. This does not only include better ones, but also worse ones, i.e. the average influence of all particles in the current group instead of global best, $Gbest$, denoted as $\overline{X}_j(t) = \frac{\sum_{i=1}^{N/K} X_{ij}}{N/K}$. α is the group influence factor. It has been found that neighbor control is able to increase the swarm diversity, which causes an improvement in the performance of PSO[20].

2.3.3 Computational complexity

According to the descriptions and definitions above, the pseudo code of the IPSO algorithm can

be summarized in Algorithm 1. We can see that the IPSO is as simple as the Classic PSO. In Algorithm 1, the largest computational cost is the update of velocity and position of each particle. Therefore, the computational complexity is $O(2NM)$, where N is the number of particles in the swarm and M is the searching dimensionality.

Algorithm 1: The pseudo code of the Improved PSO.

N is the number of particles in a swarm, and each particle has M dimensions. K is the number of groups and the size of each group is N/K. $X_{i,j}$ denotes the jth particle in ith group, and t is the number of generations.

```
t=0;
Create and initialize a swarm P(t);
repeat
    Fitness evaluation and sorted according to an increasing order;
    All particles mode K to K groups;
    for each group i ∈ [1,2, … , K] do
        U = ∅
        The best particle X_{i,0} into U;
        for each particle j ∈ [1,2, … , group size] do
            Perform velocity and position update according to (6) and (7) for X_{i,j};
            Add update X_{i,j} into U;
        end
        while U ≠ ∅ do
            Randomly choose two particles X_{i,1}(t), X_{i,2}(t) from U;
            Crossover operator on X_{i,1}(t), X_{i,2}(t) and generate X_{i,1}(t + 1),
            X_{i,2}(t + 1) to P(t + 1);
            Remove X_{i,1}(t), X_{i,2}(t) from U;
        end while
    end
    t=t+1;
until termination condition is met;
```

3 Results and discussions

In order to investigate the feasibility of our method, we performed a set of tests of increasing scale using real gene expression time series data[21]. The data are 5080 genes expression profiles across 48 individual 1-hour timepoints from the intraerythrocytic developmental cycle of plasmodium falciparum using the DNA microarray, which illustrated an intimate relationship between transcriptional regulation and the developmental progression of this highly specialized parasitic organism.

Usually, the first step to analyze gene expression data requires the use of clustering techniques, which is essential in the data mining process to reveal natural structures and identify interesting patterns in the underlying data[22, 23]. So the K-mean was used to divide all the genes into 200 clusters. We tested six clusters with the size of 5, 8, 10, 15, 20 and 30 genes per network and their searching dimensionalities were 25, 64, 100, 225, 400 and 900 respectively. The first three tests can be thought of as low-dimensional problems and final three tests as high-dimensional problems.

All the experiments were done using a computer with an Intel i5 2.6GHz processor and 8GB of memory. The operating system used was OS X 10.9.5. The algorithm was implemented in Java. All experimental results were obtained from 20 independent runs.

The performance of IPSO is dependent of the parameter selection. The inertia weight $\omega_t = \frac{\omega_{max} - t/max\_iteration}{\omega_{max} - \omega_{min}}$ will become smaller with an increase in the number of iterations, where $\omega_{max} = 1$ and $\omega_{min} = 0$. Large values of ω_t facilitate global exploration while smaller values encourage a local search. c_1 and c_2 are known as the cognitive and social components and are usually fixed. In the paper, $c_1 = 0.5$ and $c_2 = 0.5$. $\varphi_1(t)$ and $\varphi_2(t)$ are two randomly generated numbers with [0,1], and α is the group influence factor, so the value is small; $\alpha = 0.01$. Also, the maximum number of iterations is 100 and the swarm size is 1000. A large swarm is good for improving the performance in high-dimensional problems. The dimensionality of the particle depends on the number of genes per network. Finally, $P_{crossover} = 0.2$.

3.1 Tests on a different number of groups

Firstly, we tested the influence of a different number of groups. The number of 15 genes per network was chosen as an example, because this is almost the median in these six experiments. Figure 2 shows the result of the different number of groups in 15 genes per network. From the Figure 2, we can see that the best result is 100 groups in a swarm. So we chose the group number equal to 100 in this paper.

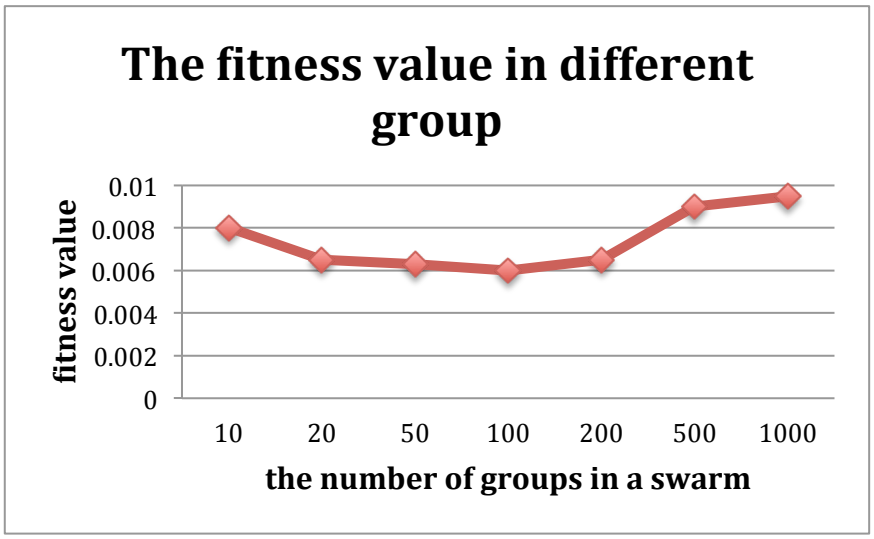


Fig. 2 The fitness value of the different number of groups in 15 genes per network.

3.2 Performance on low-dimensional GRNs

Firstly, we tested the performance of IPSO on low-dimensional GRNs. There are three different gene networks; those having 5, 8 and 10 genes. The dimensionality of the particle is 25, 64 and 100. In the past, researchers tested GRNs on a small size network. Table 1 shows the IPSO and PSO results on small size GRNs. We can see that IPSO and PSO both have good results on small size GRNs and IPSO has better results than PSO, however IPSO spends a little more computational time than PSO.

Table 1. The result of IPSO and PSO on low-dimensional GRNs.

	dimensionality of particle	25	64	100
IPSO	fitness value of training	0.0025	0.0037	0.0051
	fitness value of testing	0.0039	0.0052	0.0055
	running time (seconds)	3.5	5.1	6.6
PSO	fitness value of training	0.0041	0.0051	0.0057
	fitness value of testing	0.0055	0.0058	0.0061
	running time (seconds)	2.3	3.1	3.5

3.3 Performance on high-dimensional GRNs

In the optimization of low-dimensional GRNs, IPSO has shown good performance. However, we are keen to further test its performance on high-dimensional (large-scale) GRNs, which usually have higher than 100 searching dimensionality. Table 2 demonstrates the result of IPSO and PSO on high-dimensional GRNs. From Table 2, we can see that IPSO has decent results for high-dimensional GRNs. Even with the increase in dimensionality, IPSO also has a good and stable result. However, with the increase of dimensionality, the fitness value of PSO increases very fast. For high-dimensional GRNs, the running time of IPSO and PSO is almost the same.

	dimensionality of particle	225	400	900
IPSO	fitness value of training	0.0041	0.0047	0.0071
	fitness value of testing	0.0053	0.0061	0.0092
	running time (seconds)	64	118	260
PSO	fitness value of training	0.0061	0.0083	0.037
	fitness value of testing	0.0072	0.013	0.052
	running time (seconds)	60	107	245

4 Conclusions

In this paper, we have introduced an improved PSO approach to solve the inference problem in large-scale gene regulatory networks using differential equations. Three aspects were used to increase the diversity of swarm. Our method has been shown to work consistently well on six test examples with the search dimensional varying from 25 to 900. We obtained satisfactory results that converge in a reasonable time. In the future, we would like to investigate other real-world problems using our method and how to infer more large-scale gene regulatory networks.

Acknowledgements

The work was supported by Foundation Franco-Chinoise Pour La Science Et Ses Applications (FFCSA), the National Natural Science Foundation of China under Grants 61571312 and 61201438, the Returned Overseas Chinese Scholars Project of Education Ministry of China (20111139), the Science and Technology Support Project of Sichuan Province of China (2011GZ0201, and 2013SZ0071). Yongqing Zhang was supported by China Scholarship Council (201306240048).

References

[1] M. Bansal, V. Belcastro, A. Ambesi‐Impiombato, and D. Di Bernardo, "How to infer gene networks from expression profiles," *Molecular systems biology,* vol. 3, p. 78, 2007.

[2] Y. F. Leung and D. Cavalieri, "Fundamentals of cDNA microarray data analysis," *TRENDS in Genetics,* vol. 19, pp. 649-659, 2003.

[3] H. Huang, C.-C. Liu, and X. J. Zhou, "Bayesian approach to transforming public gene expression repositories into disease diagnosis databases," *Proceedings of the National Academy of Sciences,* vol. 107, pp. 6823-6828, 2010.

[4] R. Pinho, V. Garcia, M. Irimia, and M. W. Feldman, "Stability depends on positive autoregulation in Boolean gene regulatory networks," *PLoS Comput Biol,* vol. 10, p. e1003916, 2014.

[5] F. Dondelinger, S. Lèbre, and D. Husmeier, "Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure," *Machine Learning,* vol. 90, pp. 191-230, 2013.

[6] N. Noman, L. Palafox, and H. Iba, "Reconstruction of gene regulatory networks from gene expression data using decoupled recurrent neural network model," in *Natural Computing and Beyond*, ed: Springer, 2013, pp. 93-103.

[7] L. Palafox, N. Noman, and H. Iba, "Reverse engineering of gene regulatory networks using dissipative particle swarm optimization," *Evolutionary Computation, IEEE Transactions on,* vol. 17, pp. 577-587, 2013.

[8] X. Cai, J. A. Bazerque, and G. B. Giannakis, "Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations," *PLoS Comput Biol,* vol. 9, p. e1003068, 2013.

[9] G. A. Ruz and E. Goles, "Learning gene regulatory networks using the bees algorithm," *Neural Computing and Applications,* vol. 22, pp. 63-70, 2013.

[10] R. Xu, G. K. Venayagamoorthy, and D. C. Wunsch, "Modeling of gene regulatory networks with hybrid differential evolution and particle swarm optimization," *Neural Networks,* vol. 20, pp. 917-927, 2007.

[11] R. C. Eberhart and Y. Shi, "Particle swarm optimization: developments, applications and resources," in *evolutionary computation, 2001. Proceedings of the 2001 Congress on*, 2001, pp. 81-86.

[12] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences,* vol. 291, pp. 43-60, 2015.

[13] W. Xian, B. Long, M. Li, and H. Wang, "Prognostics of lithium-ion batteries based on the verhulst model, particle swarm optimization and particle filter," *Instrumentation and Measurement, IEEE Transactions on,* vol. 63, pp. 2-17, 2014.

[14] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *Cybernetics, IEEE Transactions on,* vol. 43, pp. 1656-1671, 2013.

[15] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *Evolutionary Computation, IEEE Transactions on,* vol. 18, pp. 82-97, 2014.

[16] A. A. Esmin, R. A. Coelho, and S. Matwin, "A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data," *Artificial Intelligence*

Review, vol. 44, pp. 23-45, 2015.

[17] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *ICML*, 1997, pp. 412-420.

[18] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *Evolutionary Computation, IEEE Transactions on,* vol. 8, pp. 225-239, 2004.

[19] A. Noor, E. Serpedin, M. Nounou, and H. Nounou, "Inferring gene regulatory networks via nonlinear state-space models and exploiting sparsity," *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB),* vol. 9, pp. 1203-1211, 2012.

[20] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *Evolutionary Computation, IEEE Transactions on,* vol. 10, pp. 281-295, 2006.

[21] Z. Bozdech, M. Llinás, B. L. Pulliam, E. D. Wong, J. Zhu, and J. L. DeRisi, "The transcriptome of the intraerythrocytic developmental cycle of Plasmodium falciparum," *PLoS Biol,* vol. 1, p. e5, 2003.

[22] D. Jiang, C. Tang, and A. Zhang, "Cluster analysis for gene expression data: a survey," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 16, pp. 1370-1386, 2004.

[23] M. F. Ramoni, P. Sebastiani, and I. S. Kohane, "Cluster analysis of gene expression dynamics," *Proceedings of the National Academy of Sciences,* vol. 99, pp. 9121-9126, 2002.

Data exchange topologies for the DISCO-HITS algorithm to solve the QAP

Omar Abdelkafi, Lhassane Idoumghar, Julien Lepagnot, and Mathieu Brévilliers

Université de Haute-Alsace (UHA)
LMIA (E.A. 3993)
4 rue des frères lumière, 68093 Mulhouse, France
{omar.abdelkafi, lhassane.idoumghar, julien.lepagnot, mathieu.Brevilliers}@uha.fr

Abstract. Exchanging information between processes in a distributed environment can be a powerful mechanism to improve results for combinatorial problem. In this study, we propose three exchange topologies for the distance cooperation hybrid iterative tabu search algorithm called DISCO-HITS. These topologies are experimented on the quadratic assignment problem. A comparison between the three topologies is performed using 21 well known instances of size between 40 and 150. Our algorithm produces competitive results and can outperform algorithms from the literature for many benchmark instances.

Keywords: Metaheuristics, DISCO-HITS, Quadratic assignment problem, Topologies.

1 Introduction

The Quadratic assignment problem (QAP) is an NP-hard problem. It is well known for its multiple applications. Many practical problems in electronic, chemistry, transport, industry and many others can be formulated as QAP. This problem was first introduced by Koopmans and Beckmann [1] to model a facility location problem. It can be described as the problem of assigning a set of facilities to a set of locations with given distance and flow between locations and facilities, respectively. The objective is to place the facilities on locations in such a way that the sum of the products between flows and distances is minimized. The problem can be formulated as follows:

$$\min_{p \in P} z(p) = \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p(i)p(j)} \tag{1}$$

where f and d are the flow and distance matrices respectively, $p \in P$ represents a solution where p_i is the location assigned to facility i and P is the set of all n

vector permutations. The objective is to minimize $z(p)$, which is the total cost assignment for the permutation p.

In this work, we propose an experimental analysis of different exchanging topologies to solve the QAP. The aim of this work is to explore the influence of these topologies. The parallel level used is the algorithmic level [2].

The rest of the paper is organized as follows. In section 2, we review some of the best-known distributed approaches to solve the QAP. In section 3, we describe the different topologies used in this work. Section 4 shows the experimental results for a set of QAPLIB instances. Finally, in section 5, we conclude the paper and we propose some perspectives.

2 Background

Since its introduction in 1957 [1], the QAP became an important problem in theory and practice. It can be considered as one of the hardest combinatorial problems due to its computational complexity. Different metaheuristics have been proposed to provide competitive results [3][4][5][6][7].

The parallel and distributed design of metaheuristic approaches has the capacity to improve the solution quality and to reduce the execution time. The computational cost of the QAP and its difficult search space make this problem suitable for parallelization. The parallel and distributed design of metaheuristics to solve the QAP is underexploited. Very few works propose it, such as the Robust Tabu search (Ro-Ts) [3] which is a parallelization of neighborhood between different processors.

In 2001, a parallel model of ant colonies is proposed [8]. A central memory to manage all communications of the search information is implemented in the master process. The search information is composed of the pheromone matrix and the best solution found. At each iteration, the master broadcasts the pheromone matrix to all the ants. Each process represents one ant and each ant constructs a complete solution and applies a Tabu Search (TS) in parallel. The process sends the solution found and the local pheromone matrix to the master. The master updates the search information. In 2005, a parallel path-relinking algorithm is proposed [9]. This proposition generates different solutions by applying path-relinking to a set of trial solutions. To improve the solutions created by the path-relinking procedure, the Ro-Ts algorithm is run in parallel starting with different trial solutions. It allows the reduction of the execution time but it does not change the behavior of the sequential algorithm and the solution quality. In 2009, a cooperative parallel TS algorithm for the QAP is introduced [6]. This approach initializes as many starting solutions as there are available processors. Each processor executes one independent TS in parallel. The initialization phase provides good starting solutions while maintaining some level of diversity. After the initialization, at each iteration, all the processors execute a TS in parallel. At the end of the generation, the current processor compares its solution with its neighbor process. If the neighbor process gets better results, the current process replaces its current solution with a mutated copy of the neighbor solution. In

2015, a parallel hybrid algorithm is proposed [10]. This proposition is composed of three steps. The first step is the seed generation which consists in using a parallel Genetic Algorithm (GA) based on the island model. Each process represents an island and at each generation, the master broadcasts the global best solution to all islands. All nodes execute a GA in parallel. The second step is the TS diversification. This method is applied to all the parallel nodes. Finally, the global best solution obtained with the first two steps is used as an initial seed for the Ro-Ts.

3 Topologies to exchange information between processes

Algorithm 1 Distance Cooperation Between Hybrid Iterative Tabu Search

1: **Input**: *perturb*: % perturbation; n: size of solution; *cost*: cost of the current solution; $Fcost$: best cost found; $S_{current}$: current solution; S_{best}: best solution found; S_{EX}: solution exchanged;
2: Initialization of the solution for the current process;
3: **repeat**
4: TS algorithm; [3]
5: **if** cost < Fcost **then**
6: $Fcost = cost$;
7: Update the S_{best} with $S_{current}$;
8: **end if**
9: level = 0; counter = 0;
10: Exchange $S_{current}$ between processes;
11: **for** i = 0 to n /* Compute distances */ **do**
12: **if** $S_{current}[i] == S_{EX}[i]$ **then**
13: counter ++;
14: **end if**
15: **end for**
16: **if** counter < $\frac{n}{4}$ **then**
17: level = 0; /* Big distance between the two processes */
18: **else**
19: **if** counter < $\frac{3 \times n}{4}$ **then**
20: level = 1; /* Processes are relatively close */
21: **else**
22: level = 2; /* Processes are very close */
23: **end if**
24: **end if**
25: **if** level == 0 **then**
26: Update $S_{current}$ with the **UX** of S_{best};
27: **else**
28: **if** level == 1 **then**
29: **Perturbation** of $S_{current}$ with the **perturb** parameter;
30: **else**
31: **Re-localization** of $S_{current}$;
32: **end if**
33: **end if**
34: **until** (Stop condition)

In 2015, a cooperative Iterative Tabu Search (ITS) called DIStance COoperation between Hybrid Iterative Tabu Search (DISCO-HITS) is proposed [11]. Each process performs an ITS in which a Ro-Ts is executed at each generation. After each iteration, each process sends its current solution to the neighbor process. Then, a distance is computed between the current solution and the solution received from the neighbor process. According to this distance, the al-

gorithm takes the decision to apply the uniform crossover (UX), to perturb the solution or to make a re-localization of this solution. Algorithm 1 presents the DISCO-HITS version used in this paper.

Exchanging information between processes (Algorithm 1 line 10) is performed according to a topology. Algorithm 1 sends its current solution to one process and receives the current solution of another process. We propose three topologies in this paper. All the topologies are defined with a sequence. Process with index i sends to process with index *i+1* and receives from index *i-1*. The last index sends its information for the first index to close the circle of exchange. This method ensures the sending and receipt of only one solution.

The first topology is the classical ring architecture implemented in the variant called *DISCO-RING-UX*. Each process sends its current solution to the next process and receives from the previous process. For example, if we use four processes, the sequence of exchange is $\{0; 1; 2; 3\}$. with this sequence, process 2 sends to process 3 and process 3 sends to process 0. This sequence is constant from the beginning of the execution to the end. The aim of this topology is to experiment a constant impact between two processes.

The second topology is the random architecture implemented in the variant called *DISCO-RANDOM-UX*. Each process sends its current solution to a random process and receives from a random process. For example if we use four processes the sequence of exchange can be $\{1; 2; 0; 3\}$. This sequence is randomly perturbed before each exchange. The aim of this topology is to experiment a dynamic impact between two processes. The random exchange allows a better diversification.

The last topology is a learning sequence architecture based on the fast ant algorithm implemented in the variant called *DISCO-LEARNING-UX*. In this case, our ant is the sequence of exchange. If the previous sequence allows the algorithm to improve, a quantity of pheromone is deposited for the pair of processes which exchange the current solution. Otherwise, the quantity of pheromone deposited is significantly reduced. Before the exchanging step, the pheromone matrix is updated and the ant is reconstructed. After the reconstruction, a step of evaporation is performed. The aim is to learn the best topology to exchange information by converging to the best sequence.

4 Experimental results

4.1 Platform and tests

In our experimentation, the algorithm is written in C/C++. It runs on a cluster of 8 machines Intel Core processor i5-3330 CPU (3.00GHz) with 4 GB of RAM and an NVIDIA GeForce GTX680 GPU. The proposed algorithm is experimented on benchmark instances from the QAPLIB [13]. The size of the instances varies between 40 and 150. Every instance is executed 10 times and the average results of these executions are given in the experiments. All the results are expressed as a percentage deviation from the best known solutions (BKS) (eq 2).

$$deviation = \frac{(solution - BKS) \times 100}{BKS} \tag{2}$$

The QAPLIB archive comprises 136 instances that can be classified into four types: real life instances (type 1); unstructured randomly generated instances based on a uniform distribution (type 2); randomly generated instances similar to real life instances (type 3); instances in which distances are based on the Manhattan distance on a grid (type 4).

Table 1. parameter of *DISCO-HITS*

Parameters	**Value**
TSiteration	$1000 \times n$
global iteration	200
aspiration criteria	$n \times n \times 5$
percentage of perturbation	25%

Table 2. Comparison of different topologies

Instance(21)	**BKS**	**DISCO-RING-UX**		**DISCO-RANDOM-UX**		**DISCO-LEARNING-UX**	
		deviation	time	deviation	time	deviation	time
tai40a	3139370	0.067(1)	3.59	**0.059(2)**	3.4	0.067(1)	3.6
tai50a	4938796	0.317(0)	6.65	0.344(0)	6.6	**0.308(0)**	6.7
tai60a	7205962	0.401(0)	11.6	0.400(0)	11.4	**0.317(0)**	11.4
tai80a	13515450	0.605(0)	27.2	0.613(0)	27.1	**0.590(0)**	27.2
tai100a	21052466	0.493(0)	53.9	0.478(0)	53.8	**0.462(0)**	53.8
tai50b	458821517	0.000(10)	6.5	0.000(10)	6.5	0.000(10)	6.6
tai60b	608215054	0.000(10)	11.3	0.000(10)	11.2	0.000(10)	11.3
tai80b	818415043	0.000(10)	27	0.000(10)	26.9	0.000(10)	27
tai100b	1185996137	0.000(10)	53.2	0.000(10)	53	0.000(10)	53.2
tai150b	498896643	0.151(0)	190	**0.129(0)**	189	0.139(0)	196.1
sko72	66256	0.001(8)	19.6	**0.000(10)**	19.5	0.001(9)	19.7
sko81	90998	0.004(6)	28	0.004(6)	28	**0.002(8)**	28.1
sko90	115534	0.001(8)	38.5	**0.000(10)**	38.6	0.001(8)	38.6
sko100a	152002	0.005(6)	53.5	**0.004(8)**	53.5	0.005(8)	53.5
sko100b	153890	0.002(8)	53.5	**0.001(9)**	53.3	0.002(8)	53.5
sko100c	147862	0.002(1)	53.5	**0.001(6)**	53.3	0.001(2)	53.5
sko100d	149576	0.004(4)	53.5	**0.002(5)**	53.4	0.005(4)	53.5
sko100e	149150	0.002(6)	53.7	**0.002(8)**	53.3	0.002(7)	53.4
sko100f	149036	0.004(3)	53.6	0.006(3)	53.8	**0.003(4)**	53.4
wil100	273038	0.003(1)	53.6	0.003(2)	53.5	**0.002(3)**	53.6
tho150	8133398	**0.016(0)**	198.1	0.030(0)	189.3	0.021(0)	191.4
Average type 2		0.3766(1)	20.6	0.3788(2)	20.5	**0.3488(1)**	20.5
Average type 3		0.0302(40)	57.6	**0.0258(40)**	57.3	0.0278(40)	58.8
Average type 4		**0.0040(51)**	59.9	0.0048(67)	59	0.0041(61)	59.3
Average		0.099(92)	50	0.099(109)	49.5	**0.092(102)**	49.9

Table 3. Comparison with the literature

Instance(19)	**BKS**	**DISCO-RING-UX**		**DISCO-RANDOM-UX**		**DISCO-LEARNING-UX**		**TLBO-RTS**		**CPTS**	
		deviation	time	deviation	time	deviation	time	deviation	time	deviation	time
tai40a	3139370	0.067(1)	3.59	0.059(2)	3.4	0.067(1)	3.6	**0.000**	29	0.148(1)	3.5
tai50a	4938796	0.317(0)	6.65	0.344(0)	6.6	**0.308(0)**	6.7	0.360	55	0.440(0)	10.3
tai60a	7205962	0.401(0)	11.6	0.400(0)	11.4	**0.317(0)**	11.4	0.410	95.3	0.476(0)	26.4
tai80a	13515450	0.605(0)	27.2	0.613(0)	27.1	**0.590(0)**	27.2	0.870	239.5	0.691(0)	94.8
tai100a	21052466	0.493(0)	53.9	0.478(0)	53.8	**0.462(0)**	53.8	0.596	483.3	0.589(0)	261.2
tai80b	818415043	0.000(10)	27	0.000(10)	26.9	0.000(10)	27	0.000	239	0.000(10)	110.9
tai100b	1185996137	0.000(10)	53.2	0.000(10)	53	0.000(10)	53.2	0.000	508.2	0.001(8)	241
tai150b	498896643	0.151(0)	190	0.129(0)	189	0.139(0)	196.1	**0.015**	428.5	0.076(0)	7377.8
sko72	66256	0.001(8)	19.6	0.000(10)	19.5	0.001(9)	19.7	0.000	172.8	0.000(10)	69.6
sko81	90998	0.004(6)	28	0.004(6)	28	0.002(8)	28.1	0.000	348.2	0.000(10)	121.4
sko90	115534	0.001(8)	38.5	0.000(10)	38.6	0.001(8)	38.6	0.000	342.8	0.000(10)	193.7
sko100a	152002	0.005(6)	53.5	0.004(8)	53.5	0.005(8)	53.5	0.003	594.3	**0.000(10)**	304.8
sko100b	153890	0.002(8)	53.5	0.001(9)	53.3	0.002(8)	53.5	0.005	482.6	**0.000(10)**	309.6
sko100c	147862	0.002(1)	53.5	0.001(6)	53.3	0.001(2)	53.5	0.000	508.5	0.000(10)	316.1
sko100d	149576	0.004(4)	53.5	0.002(5)	53.4	0.005(4)	53.5	0.009	509.4	**0.000(10)**	309.8
sko100e	149150	0.002(6)	53.7	0.002(8)	53.3	0.002(7)	53.4	0.005	614.5	**0.000(10)**	309.1
sko100f	149036	0.004(3)	53.6	0.006(3)	53.8	0.003(4)	53.4	0.005	482.6	0.003(4)	310.3
wil100	273038	0.003(1)	53.7	0.003(2)	53.5	0.002(3)	53.6	0.000	482.6	0.000(10)	316.6
tho150	8133398	0.016(0)	198.1	0.030(0)	189.3	0.021(0)	191.4	0.030	556.6	**0.013(0)**	1991.7
Average type 2		0.3766(1)	20.6	0.3788(2)	20.5	**0.3488(1)**	20.5	0.4472	180.42	0.4688(1)	79.2
Average type 3		0.0503(20)	90	0.0430(20)	89.6	0.0463(20)	92.1	**0.0050**	391.9	0.0257(18)	2576.6
Average type 4		0.0040(51)	59.9	0.0048(67)	59	0.0041(61)	59.3	0.0052	463.2	**0.0014(94)**	413.9
Average		0.109(72)	54.3	0.109(89)	53.7	**0.101(82)**	54.3	0.121	377.5	0.128(113)	667.3
Average NOFE		1.48e+08		1.48e+08		1.48e+08		7.55e+10		9.23e+08	

4.2 Parameters

DISCO-HITS contains a set of parameters. A set of experimentation is executed to fix all the parameters. Table 1 shows the parameters used in the experimentation, where n is the size of the problem and *rank* is the index of the current process.

4.3 Experimentation of the three topologies

Table 2 contains the results for the three variants proposed in this work. The same number of objective function evaluations and the same machines are used (equivalent computing power). The time is expressed in minutes. The number within brackets is the number of times each algorithm gets the BKS among the 10 trials.

Through the 21 benchmark instances presented in this work, *DISCO-RING-UX* outperforms all the variants for only one instance (tho150 in type 2). *DISCO-RANDOM-UX* outperforms all the variants for 9 instances especially from type 4. Finally, *DISCO-LERNING-UX* outperforms all the variants for 7 instances especially from type 3. *DISCO-LERNING-UX* gets the best global average of 0.092%. This variant shows the most stable results for the 3 types.

4.4 Literature Comparison

Table 3 presents several comparisons with two distributed algorithms from the literature. Cooperative parallel tabu search (CPTS) [6] (2009) and Teaching-Learning-Based Optimization (TLBO) [12] (2015).

The average number of objective function evaluation (NOFE in Table 3) used in our 3 variants is much lower than for the literature algorithms. *CPTS* algorithm uses 5.8 times more objective function evaluations and *TLBO* uses 523.5 times more evaluations. We use 19 well-known benchmark instances from the QAPLIB which are difficult to solve. *DISCO-LERNING-UX* outperforms all the algorithms on 4 instances from type 3. *TLBO* outperforms all the algorithms on 2 instances (tai40a and tai150b). *CPTS* outperforms all the algorithms on 5 instances from type 4. *DISCO-LERNING-UX* gets the best global average of 0.101% against 0.128% for *CPTS* and 0.121% for *TLBO*. Considering the difference of NOFE, the results obtained by our 3 variants are very competitive.

5 Conclusion and perspectives

In this work, we have presented and experimented three variants of the DISCO-HITS algorithm with different topologies to solve the QAP. The results show that the proposed variants perform efficiently. We evaluated our variants on 19 benchmark instances from the QAPLIB and they get the best average results compared to two leading distributed algorithms from the literature.

In summary, the main contributions of this work are the proposition of these variants and the experimentation of three different topologies to exchange information in a distributed environment. The automatically learnt topology, used in the *DISCO-LERNING-UX* variant, shows the best average results.

In future works, there are several possible ways to extend this work. One possibility is to experiment other parameters to get better results on large neighborhood instances. An experimental analysis can also be made using some instances which are not explored in literature, such as tai729eyy. Finally, this approach can be experimented for other combinatorial problems to analyze its behavior with other kinds of problems.

References

1. T. Koopmans, M. Beckmann, Assignment problems and the location of economic activities, Econometrica, vol. 25, no. 1, pp. 53-76, 1957.
2. E.G. Talbi, Metaheuristics: from Design to Implementation, University of Lille - CNRS - INRIA, John wiley and sons Inc, 2009.
3. E. Taillard, Robust taboo search for the quadratic assignement problem, Parallel computing 17, pp. 443-455 ,1991.
4. T. James, C. Rego, F. Glover, Multistart Tabu Search and Diversification Strategies for the Quadratic Assignment Problem, IEEE TRANSACTIONS ON SYSTEMS, Man, And Cybernetics-part a: systems and humans, vol. 39, no. 3, May 2009.
5. U. Benlic, J.K. Hao, Breakout local search for the quadratic assignement problem, Applied Mathematics and Computation 219, pp. 4800-4815, 2013.
6. T. James, C. Rego, F. Glover, A cooperative parallel tabu search algorithm for the quadratic assignment problem, European Journal of Operational Research 195, pp. 810-826, 2009.
7. M. Czapinski, An effective Parallel Multistart Tabu Search for Quadratic Assignment Problem on CUDA platform, J. Parallel Distrib. Comput. 73, pp. 1461-1468, 2013.
8. E. G. Talbi, O. Roux, C. Fonlupt, D. Robillard, Parallel Ant Colonies for the quadratic assignment problem, Future Generation Computer Systems 17, pp 441-449, 2001.
9. T. James, C. Rego, F. Glover, Sequential and parallel path relinking algorithms for the quadratic assignment problem, IEEE Intelligent Systems 20 (4), pp 58-65, 2005.
10. U. Tosun, On the performance of parallel hybrid algorithms for the solution of the quadratic assignment problem, Engineering Applications of Artificial Intelligence 39, pp 267-278, 2015.
11. O. Abdelkafi, L. Idoumghar, J. Lepagnot, Comparison of Two Diversification Methods to Solve the Quadratic Assignment Problem, Procedia Computer Science 51, pp 2703-2707, 2015.
12. Tansel Dokeroglu, Hybrid teaching-learning-based optimization algorithms for the Quadratic Assignment Problem, Computers and Industrial Engineering 85, pp 86-101, 2015.
13. R.E. Burkard, S.E Karisch, F. Rendl, QAPLIB - A quadratic assignment problem library, journal of global optimization Volume: 10 Issue: 4, pp. 391-403, Jun 1997.

Distributed Local Search for Elastic Image Matching

Hongjian Wang, Abdelkhalek Mansouri, Jean-Charles Créput, Yassine Ruichek

IRTES-SeT, Université de Technologie de Belfort-Montbéliard, 90010 Belfort, France

Abstract. We propose a *distributed local search* (DLS) algorithm, which is a parallel formulation of a local search procedure in an attempt to follow the spirit of standard local search metaheuristics. Applications of different operators for solution diversification are possible in a similar way to *variable neighborhood search*. We formulate a general energy function to be equivalent to elastic image matching problems. A specific example application is *stereo matching*. Experimental results show that the GPU implementation of DLS seems to be the only method that provides an increasing acceleration factor as the instance size augments, among eight tested energy minimization algorithms.

Key words: Parallel and distributed computing, Variable neighborhood search, Stereo matching, Graphics processing unit

1 Introduction

Local search, also referred as hill climbing, descent, iterative improvement, general single-solution based metaheuristics and so on, is a metaheuristic algorithm [1]. Starting with a given initial solution, at each iteration the heuristic replaces the current solution by a neighbor solution that improves the fitness function. The search stops when all candidate neighbors are worse than the current solution, meaning a local optimum is reached. Existing parallelization strategies for local search can be divided into three categories. In the first category, the evaluation of neighborhood is made in parallel [2, 3]; in the second category, the focus is on the parallel evaluation of a single solution, and the function can be viewed as an aggregation of partial functions [2, 4]; in the third category, several local search metaheuristics are simultaneously launched for computing robust solutions [5, 6]. In our opinion, an interesting parallel implementation model of local search should be fully distributed, where each processor carries out its own neighborhood search based on some parts of the input data, considering only a local part of the whole solution. Operations on different processors should be similar, with no centralized selection procedure, except for final evaluation. A final solution should be obtained with the partial operations from different processors. Following this idea, we propose a *distributed local search* (DLS) algorithm and implement it on GPU parallel computing platforms.

A natural field of applications with GPU processing is image processing, which is a domain at the origin of GPU development. A lot of image processing

and computer vision problems can be viewed as optimization problems in a more general way, dealing with brute data distributed in some Euclidean space and system in relation to the data. More often, these NP-hard optimization problems involve data distributed in the plane and elastic structures represented by graphs that must match the data. Such optimization problems can be stated in a generic framework of *graph matching* [7,8]. In this paper, we are particularly interested in moving grids in the plane following the idea of visual correspondence problem, which is to compute the pairs of pixels from two images that result from the same scene element. A typical example application is *stereo matching*, which we formulate as an elastic image matching problem [9]. We apply the proposed DLS algorithm to stereo matching by minimizing the corresponding energy function.

The DLS can be used for parallel implementation of elastic matching problems that include not only visual correspondence problems but also neural network topological maps, or elastic nets approaches [10,11], modeling the behavior of interacting components inspired by biological systems and collective behaviors at a low level of granularity. The framework is based on data decomposition, with the idea of modeling the geometry of objects using some adaptive (elastic) structures that move in space and continuously interact with the input data distribution memorized into a cellular matrix [12]. Then spatial metaphors, as well as biological metaphors should fit well into the cellular matrix framework.

2 Elastic Grid Matching

We define a class of visual correspondence problems as *elastic grid matching* problems. Given two input images with same size and same regular topology, one is a matcher grid $G_1 = (V_1, E_1)$ where a vertex is a pixel with a variable location in the plane, while the other is a matched grid $G_2 = (V_2, E_2)$ where vertices are pixels located in a regular grid. The goal of elastic grid matching is to find the matcher vertex locations in the plane, so that the following energy function

$$E(G_1) = \sum_{p \in V_1} D_p(p - p_0) + \lambda \cdot \sum_{\{p,q\} \in E_1} V_{p,q}(p - p_0, q - q_0) \tag{1}$$

is minimized, where p_0 and q_0 are the default locations of p and q respectively in a regular grid. Here, D_p is the data energy that measures how much assigning label f_p to pixel p disagrees with the data, and $V_{p,q}$ is the smoothness energy that expresses smoothness constraints on the labelings enforcing spatial coherence [13–15]. A label f_p in visual correspondence represents a pixel moving from its regular position into the direction of its homologous pixel, *i.e.* $f_p = p - p_0$. In the following sections, we will directly use the notations of labels as relative displacements, as usual with such problems. The energy function is commonly used for visual correspondence problems, and it can be justified in terms of maximum a posteriori estimation of a *Markov random field* (MRF) [16,17].

It has been proven that elastic image matching is NP-complete [9], and finding the global minimum for the energy function even with the simplest smooth-

ness penalty, the piecewise constant prior, is NP-hard [13, 14]. We choose the local search metaheuristics to deal with the energy minimization problem.

3 Distributed Local Search

Based on the cellular matrix model proposed in [12], we design a parallel local search algorithm, called *distributed local search* (DLS), to implement many local search operations on different parts of the data in a distributed way. It is a parallel formulation of local search procedures in an attempt to follow the spirit of standard local search metaheuristics. Starting from its location in the cellular matrix, each processor locally acts on the data located in the corresponding cell according to the cellular decomposition, in order to achieve local evaluation, perform neighborhood search, and select local improvement moves to execute. The many processes locally interact in the plane, making evolve the current solution into an improved one. The solution results from the many independent local search operations simultaneously performed on the distributed data in the plane. Normally, a local search algorithm with single operator obtains local minima. In order to escape from local minima, we design several operators. Applications of different operators for diversification are possible in a similar way to the *variable neighborhood search* (VNS).

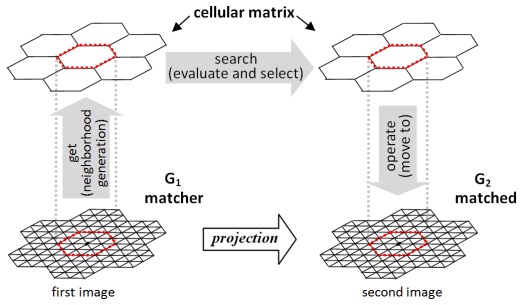


Fig. 1: Basic projection for DLS.

3.1 Data Structures and Basic Operations

The data structures and direction of operations for DLS algorithms are illustrated in Figure 1. The input data set is deployed on the low level of both matcher grid and matched grid, represented as regular images in the figure. The honeycomb cells represent the cellular matrix level of operations. Each cell is a basic processor that handles a basic local search processing iteration with the three following steps: neighborhood generation step (**get**); neighbor solution evaluation and selecting the best neighbor (**search**); then moving the matcher

grid toward the selected neighbor solution (**operate**). The nature and size of specific moves and neighborhoods will depend on the type of operator used and the level of the cellular matrix. The higher is the level, the larger is the local cell/neighborhood. In the cellular matrix model, a solution is composed of many sub-solutions from many cells. Each sub-solution is evolved from an initial sub-solution based on the distributed data in a cell. By partitioning the data and solution, the neighborhood structure is also partitioned at the same time.

3.2 Local Evaluation with Mutual Exclusion

During the parallel operation, the coherence of local evaluation with mutual exclusion is violated by conflict operations. A conflict operation occurs when a same pixel or two neighboring pixels is/are being evaluated and moved simultaneously by two threads. Conflict operations only happen on frontier pixels, which are the pixels on the cell frontiers according to the cellular matrix partition of the image. In order to eliminate the conflict operations in DLS, we propose a strategy, called *dynamic change of cell frontiers* (DCCF), by which we limit the move to the internal pixels of a cell only. Cell frontier pixels remain at fixed locations, and they are not concerned by local moves so that exclusive access of the thread to its internal region delimited by the cell, is guaranteed. A problem that arises is how to manage cell frontier pixels and make them participating in the optimization process. As a solution, the cellular matrix decomposition is dynamically changeable from the CPU side before the application of a round of DLS operations. At different moments, the cellular matrix decomposition slightly shifts on the input image in order to change the cell frontiers and consequently the fixed pixels. For a given cellular matrix decomposition, cell frontier pixels are then fixed and not allowed to be moved by current DLS operations.

3.3 Neighborhood Operators

We design different neighborhood operators for the DLS algorithm applied to the elastic grid matching. We use the notations of labeling problems to present these operators. Move operations in a given neighborhood structure correspond to changing labels of pixels in the corresponding labeling space. Operators are classified between small moves and large moves. In the first category, only a single pixel from the cell moves at a time, meaning that only one pixel's label is changed. We designed two small move operators: *local move operator* and *propagation operator*. In the second category, larger sets of pixels from a cell can simultaneously move. We designed six large move operators: *random pixels move operator*, *random pixels jump operator*, *random pixels expansion operator*, *random pixels swap operator*, *random window move operator* and *random window jump operator*. Details about these operators can be found in [12].

3.4 GPU Implementation Under VNS Framework

We use Compute Unified Device Architecture (CUDA) to implement the DLS algorithm on GPU platforms. The CUDA kernel calling sequence from the CPU

side enables the application of different operators in the spirit of VNS and manages dynamic changes of cellular matrix frontiers. According to our previous experiments, the repartition of tasks between host (CPU) and device (GPU) is actually the best compromise we found to exploit the GPU CUDA platform at a reasonable level of computation granularity. Data transfer between CPU side and GPU side only occurs at the beginning and the end of the algorithm. It is the CPU side that controls DLS kernel calls with different operators executed within the dynamic change of cell frontiers (DCCF) pattern for frontier cells management. With several neighborhood operators in hand, we use them under the VNS framework in order to enhance the solution diversification.

4 Experimental Study

We apply the DLS algorithm to stereo matching, viewing the problem as energy minimization problem. We follow in the footsteps of Boykov et al. [14], Tappen and Freeman [18], and Szeliski et al. [15], using a simple energy function, applied to benchmark images from the widely used Middlebury stereo data set [19]. The labels are the disparities, and the data costs are the absolute color differences between corresponding pixels for each disparity. For the smoothness term in the energy function, we use a truncated linear cost as the piecewise smooth prior defined in [13]. We focus on the performance of DLS when input size augments. We experiment on the Middlebury 2005 stereo benchmark [19] including 18 pairs of images with sizes from the smallest 458×370 to the largest 1374×1110 in average. We uniformly set the disparity range to 64 pixels, for all the sizes. We denote our DLS GPU implementation as DLS-gpu. We also test the counterpart CPU sequential version which is denoted by DLS-cpu. We compare DLS with six other methods[1]: *iterated conditional modes* (ICM) [16] which is an old approach using a deterministic "greedy" strategy to find a local minimum; sequential *tree-reweighted message passing* (TRW-S) [15] which is an improved version of the original tree-reweighted message passing algorithm [20]; BP-S and BP-M [15] which are two updated version of the max-product *loopy belief propagation* (LBP) implementation of [18]; GC-swap and GC-expansion which are two graph cuts based algorithms proposed in [14]. Instead of reporting the absolute energy values, we report the percentage deviation from the best known solution (lowest energy) of the mean solution value over 10 runs, denoted as $\%PDM$ value. We choose the best known solution from the executions of all tested methods.

The results of different methods are reported in Figure 2. From top to bottom are reported the energy value as $\%PDM$, the execution time, and the acceleration factor of each method relative to the slowest method (DLS-cpu) and the method (GC-expansion) that gets the lowest energy, respectively. The ICM method runs fastest but generates very high energies, while DLS-gpu runs

[1] For all the tested energy minimization algorithms, we use the original codes from *http://vision.middlebury.edu/MRF/code/*.

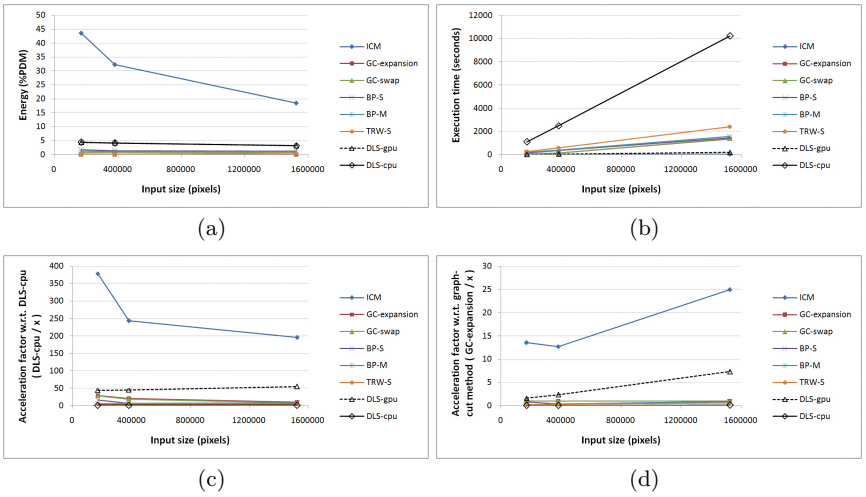


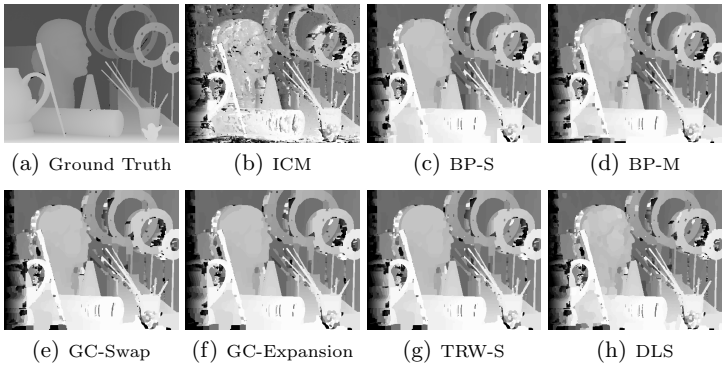
Fig. 2: Results of eight tested methods. Left column: results with different input sizes. Right column: results with different disparity ranges.

a little slower than ICM but generates much lower energies with more acceptable $\%PDM$ values smaller than 5%. An important observation from Figure 2 is that, among all the tested methods, only the DLS-gpu has an acceleration factor which increases according to the augmentation of input size. This means that further improvement could be carried on only by the use of multi-processor platform with more effective cores.

In Figure 3 are displayed the disparity maps for the *Art* benchmark. Note that during our experiments, we choose the stereo matching application but only view it as an energy minimization problem, just focusing on minimizing energies. The disparity maps obtained from all the tested methods are the raw results after energy minimization, without any additional post-treatments such as left-right consistency check, occlusion detection, or disparity smoothing, which are all treatments specific to stereo matching in order to minimize the errors compared with ground truth disparity maps. Moreover, as pointed out in [15], the ground truth solution may not always be strictly related to the lowest energy.

5 Conclusion

We have proposed a parallel formulation of local search procedure, called distributed local search (DLS) algorithm. We have applied the algorithm to stereo matching problem. The main encouraging result is that the GPU implementation of DLS on stereo matching seems to be the only method that provides an increasing acceleration factor as the instance size augments, for a result of quality



(a) Ground Truth (b) ICM (c) BP-S (d) BP-M

(e) GC-Swap (f) GC-Expansion (g) TRW-S (h) DLS

Fig. 3: Disparity maps for the *Art* (463×370) benchmark obtained with different energy minimization methods. The disparity range is set to 64 pixels.

less than 5% deviation to the best known energy value. For all the other approaches, the acceleration factor, against the slowest sequential version of DLS, is decreasing, except for the ICM method, which however only produces poor result of about 45% deviation to the best known energy. Graph cuts based algorithms and belief propagation based algorithms are well-performing approaches concerning quality, however the computation time increases quickly along with the instance size. That is why we hope for further improvements or improved accelerations of the DLS approach with the availability of new multi-processor platforms with more independent cores.

It is a well-known fact that the minimum energy level does not necessarily correlate to the best real-case matching. Here, we only address energy minimization discarding too much complex post-treatments necessary for the "true" ground truth matching. It should follow that many tricks are certainly not yet implemented to make energy minimization coincide to ground truth evaluation. In order to improve the matching quality in terms of minimizing the errors to ground truth only, specially designed terms for detecting typical situations in vision, such as occlusion, slanted surfaces, and the aperture problem, need to be added in the formulation of energy function. Furthermore, more complex post-treatments for invalid flow value fixing and smoothing should also be considered.

References

1. Talbi, E.G.: Metaheuristics: from design to implementation. Volume 74. John Wiley & Sons (2009)
2. Van Luong, T., Melab, N., Talbi, E.G.: Gpu computing for parallel local search metaheuristic algorithms. Computers, IEEE Transactions on **62** (2013) 173–185
3. Delévacq, A., Delisle, P., Krajecki, M.: Parallel gpu implementation of iterated local search for the travelling salesman problem. In: Learning and Intelligent Optimization. Springer (2012) 372–377

4. Fosin, J., Davidović, D., Carić, T.: A gpu implementation of local search operators for symmetric travelling salesman problem. PROMET-Traffic&Transportation **25** (2013) 225–234
5. Melab, N., Talbi, E.G., et al.: Gpu-based multi-start local search algorithms. In: Learning and Intelligent Optimization. Springer (2011) 321–335
6. Sánchez-Oro, J., Sevaux, M., Rossi, A., Martí, R., Duarte, A.: Solving dynamic memory allocation problems in embedded systems with parallel variable neighborhood search strategies. Electronic Notes in Discrete Mathematics **47** (2015) 85–92
7. Bengoetxea, E.: Inexact Graph Matching Using Estimation of Distribution Algorithms. PhD thesis, Ecole Nationale Supérieure des Télécommunications, Paris, France (2002)
8. Caetano, T.S., McAuley, J.J., Cheng, L., Le, Q.V., Smola, A.J.: Learning graph matching. Pattern Analysis and Machine Intelligence, IEEE Transactions on **31** (2009) 1048–1058
9. Keysers, D., Unger, W.: Elastic image matching is np-complete. Pattern Recognition Letters **24** (2003) 445–453
10. Durbin, R., Willshaw, D.: An analogue approach to the travelling salesman problem using an elastic net method. Nature **326** (1987) 689–691
11. Créput, J.C., Hajjam, A., Koukam, A., Kuhn, O.: Self-organizing maps in population based metaheuristic to the dynamic vehicle routing problem. Journal of Combinatorial Optimization **24** (2012) 437–458
12. Wang, H.: Cellular matrix for parallel k-means and local search to Euclidean grid matching. PhD thesis, Université de Technologie de Belfort-Montbeliard (2015)
13. Veksler, O.: Efficient graph-based energy minimization methods in computer vision. PhD thesis, Cornell University (1999)
14. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. Pattern Analysis and Machine Intelligence, IEEE Transactions on **23** (2001) 1222–1239
15. Szeliski, R., Zabih, R., Scharstein, D., Veksler, O., Kolmogorov, V., Agarwala, A., Tappen, M., Rother, C.: A comparative study of energy minimization methods for markov random fields with smoothness-based priors. Pattern Analysis and Machine Intelligence, IEEE Transactions on **30** (2008) 1068–1080
16. Besag, J.: On the statistical analysis of dirty pictures. Journal of the Royal Statistical Society. Series B (Methodological) (1986) 259–302
17. Geman, S., Geman, D.: Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. Pattern Analysis and Machine Intelligence, IEEE Transactions on (1984) 721–741
18. Tappen, M.F., Freeman, W.T.: Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In: Computer Vision, 2003 Ninth IEEE International Conference on, IEEE (2003) 900–906
19. Scharstein, D., Szeliski, R.: High-accuracy stereo depth maps using structured light. In: Computer Vision and Pattern Recognition, 2003 IEEE Conference on. Volume 1., IEEE (2003) I–195
20. Wainwright, M.J., Jaakkola, T.S., Willsky, A.S.: Map estimation via agreement on trees: message-passing and linear programming. Information Theory, IEEE Transactions on **51** (2005) 3697–3717

Fast Hybrid BSA-DE-SA Algorithm on GPU

Mathieu Brévilliers, Omar Abdelkafi,
Julien Lepagnot, and Lhassane Idoumghar

Université de Haute-Alsace (UHA), LMIA (E.A. 3993)
4 rue des frères Lumière, 68093 Mulhouse, France
{mathieu.brevilliers,omar.abdelkafi,
julien.lepagnot,lhassane.idoumghar}@uha.fr

Abstract. This paper introduces a hybridization of Backtracking Search Optimization Algorithm (BSA) with Differential Evolution (DE) and Simulated Annealing (SA). An experimental study, conducted on 13 benchmark problems, shows that this approach outperforms BSA in terms of solution quality and convergence speed. We also describe our CUDA implementation of this algorithm for graphics processing unit (GPU). Experimental results are reported for high-dimension benchmark problems, and it highlights that significant speedup can be achieved.

Keywords: continuous optimization, hybrid metaheuristic, backtracking search optimization algorithm, differential evolution, simulated annealing, graphics processing unit, CUDA.

1 Introduction

Evolutionary algorithms are metaheuristics that use evolution mechanisms in order to approximate the best solution of a given optimization problem. In this category, several efficient approaches have emerged, such as particle swarm optimization algorithms or differential evolution algorithms. Among all existing evolutionary strategies, the Backtracking Search Optimization Algorithm (BSA) [2] can also find high-quality solutions for continuous optimization problems, and several extensions have been proposed to improve either solution quality or convergence speed [1, 3, 7]. As BSA mainly focuses on exploration, it can be quite slow converging on the global best solution, and it would be challenging to speed up its convergence without loss of quality.

To this aim, we present a hybrid algorithm that uses differential evolution (DE) and simulated annealing (SA) techniques together with BSA principles. We also propose an implementation for graphics processing unit (GPU) to investigate the benefit in terms of runtime speedup for high-dimension instances.

Section 2 presents BSA and two BSA-DE hybridizations from the literature. Section 3 introduces our BSA-DE-SA hybrid approach and reports experimental results. The corresponding GPU design is described in Section 4, and an experimental study shows to what extent the algorithm can be accelerated. Finally, concluding remarks and perspectives are given in Section 5.

2 Related work

2.1 Backtracking search optimization algorithm

Backtracking Search Optimization Algorithm (BSA) [2] is an evolutionary algorithm for continuous optimization. BSA is based on a population evolving with classical operators: mutation, crossover, boundary control, and selection. However, as a backtracking strategy, BSA has a memory to store a historical population, that consists of the individuals of a previous generation. Before applying the mutation operator, this memory is updated with probability 0.5, by replacing the whole historical population with a random permutation of the current population. Then, a new mutant population M is created from the current population P and from the historical population $oldP$ by using the following equation:

$$\forall i \in \{1, ..., N\}, \forall j \in \{1, ..., D\}, M_{i,j} = P_{i,j} + F^{BSA} \times (oldP_{i,j} - P_{i,j}) \quad (1)$$

where N is the number of individuals in P, D is the number of dimensions in the considered optimization problem, $F^{BSA} = 3 \times randn$, and $randn$ is a real value randomly generated with the standard normal distribution. A new value of F^{BSA} is generated for each generation.

A first advantage of BSA is that it has few user-defined parameters: the population size N, and a so-called $mixrate$ parameter that controls how many dimensions (at most) of a mutant individual will be incorporated in a trial individual after the crossover. Moreover, BSA can solve a wide range of optimization problems, due to its good exploration ability, and it has been shown [2] that it performs better than SPSO2011, CMAES, ABC, JDE, CLPSO, and SADE.

2.2 Hybrid BSA-DE algorithms

We present here two hybridizations that inspired the algorithm proposed in Section 3. Firstly, Das et al.[3] replaced Equation 1 of BSA in the following way:

$$\forall i \in \{1, ..., N\}, \forall j \in \{1, ..., D\},$$
$$M_{i,j} = P_{i,j} + F^{BSA} \times (oldP_{i,j} - P_{i,j}) + F^{DE} \times (P_{best,j} - P_{i,j}) \quad (2)$$

where F^{BSA} is defined as in Equation 1, F^{DE} is the scaling factor of DE, and $best \in \{1, ..., N\}$ is the index of the best individual in P. In contrast with BSA, a new value of F^{BSA} is generated for each individual. It has been shown that this BSA-DE hybridization generally performs better than BSA, and converges faster than BSA and DE.

Wang et al.[7] proposed a hybridization where DE follows BSA in the generation loop: DE is applied to improve only 1 bad individual of the current population. This bad individual is randomly chosen with respect to its fitness: the worse the fitness, the higher the probability. Then, the DE/best/1 mutation scheme and a binomial crossover are used to generate a trial individual, that will replace the current individual if it performs better. Comparing this so-called HBD algorithm with BSA, it has been shown that HBD outperforms BSA in terms of solution quality and convergence speed.

3 Contribution to speed up BSA convergence

The proposed hybrid approach is based on a two-level BSA-DE combination and on a SA schedule to gradually decrease the range of BSA scaling factor. The aim is to improve the convergence of the basic BSA algorithm.

Individual-level BSA-DE hybridization. We define 2 new scaling factors. The first one, called *intensification factor*, and denoted F^I, is defined by the user in $[0, 1]$. The second one, called *exploration factor*, and denoted F_i^E, is generated for each individual i during the mutation process: $\forall i \in \{1, ..., N\}, F_i^E = C \times randn$, where C is a coefficient decreasing with time (see below). Then, Equation 1 is modified as follows, in a slightly different way from [3], in order to instill the DE/target-to-best/1 scheme into BSA mutation operator:

$$\forall i \in \{1, ..., N\}, \forall j \in \{1, ..., D\},$$
$$M_{i,j} = P_{i,j} + F_i \times (oldP_{i,j} - oldP_{k,j}) + F^{DE} \times (P_{best,j} - P_{i,j}), \quad (3)$$

where k is randomly chosen in $\{1, ..., N\}$ such that $k \neq i$. The factor F_i replaces F^{BSA}, and is defined by the equation:

$$F_i = \begin{cases} F_i^E & \text{if } rand > \frac{1}{16}, \\ F^I & \text{otherwise,} \end{cases} \quad (4)$$

where $rand$ is a random value uniformly generated in $[0, 1]$.

SA schedule for C. According to the temperature cooling schedule in SA, the coefficient C is gradually decreased from 3 to 1 with a geometric law during the first third of the algorithm (in terms of number of function evaluations).

Generation-level BSA-DE hybridization. The method proposed in [7] is applied after each iteration of the individual-level BSA-DE hybridization.

Equation 4 together with the range of C and F^I show that a few individuals are used to intensify the search with a low F_i, while the major part explores the search space with a larger F_i. Furthermore, the SA schedule for decreasing C allows to use the full exploration ability of the algorithm at the beginning, and to develop its exploitation ability at a later stage. Finally, the two-level BSA-DE hybridization allows to combine in the same algorithm the DE/best/1 scheme (generation-level) with a DE/target-to-best/1-like scheme (individual-level), in order to speed up the convergence of the algorithm.

We realized an experimental study in order to compare our hybrid BSA-DE-SA approach with BSA [2], BSA-DE [3], and HBD [7]. Specifically, two versions of BSA-DE-SA have been implemented: BDS-1 that only uses the individual-level BSA-DE hybridization with a SA schedule for C, and BDS-2 that uses all features described above. All these algorithms have been tested on the benchmark functions listed in Table 1, and Table 2 shows the values of the control parameters for each algorithm. Each algorithm has been run 30 times on each benchmark function. $10\,000 \times D$ function evaluations per run are allowed, and a benchmark problem is considered as solved when a fitness lower than $f_{opt} + 10^{-8}$ is reached, where f_{opt} denotes the corresponding optimal fitness.

Table 1. List of benchmark problems (ID: function identifier; Low, Up: limits of search space; D: dimension).

ID	Name	Low	Up	D
F1	Schwefel 1.2	-100	100	30
F2	Ackley	-32	32	30
F3	Rastrigin	-5.12	5.12	30
F4	Rosenbrock	-30	30	30
F5	Weierstrass	-0.5	0.5	10
F6	Shifted Schwefel 1.2	-100	100	10
F7	Shifted rotated high conditioned elliptic function	-100	100	10
F8	Shifted Schwefel 1.2 with noise	-100	100	10
F9	Schwefel 2.6	-100	100	10
F10	Shifted Rosenbrock	-100	100	10
F11	Shifted rotated Griewank	0	600	10
F12	Shifted rotated Ackley	-32	32	10
F13	Shifted Rastrigin	-5	5	10

Table 2. Control parameter settings for the compared algorithms.

Algorithm	Parameters
BSA [2]	$N = 30$, $mixrate = 1$.
BSA-DE [3]	$N = 30$, $mixrate = 1$, $F^{DE} = 0.5$.
HBD [7]	$N = 30$, $mixrate = 1$, scaling factor $F = 0.8$, crossover rate $C_r = 0.9$, DE applied on $N/30 = 1$ individual.
BDS-1	$N = 30$, $mixrate = 1$, $F^{DE} = 0.5$, $F^I = 0.5$ applied for each individual with probability 1/16, C decreased from 3 to 1 during the first 1/3 of the allowed function evaluations.
BDS-2	BDS-1 settings together with HBD settings.

Table 3 reports basic statistics for the compared algorithms. We can see that BDS-2 gets 10 times the first place in terms of mean error, whereas BSA-DE, BDS-1, HBD and BSA make it respectively 9, 8, 6, and 4 times. BDS-2 beats BSA on 9 functions (F1, F3, F4, F6-11), HBD on 6 functions (F1, F3, F7, F9, F10, F12), and BSA-DE on 4 functions (F7, F10-12). Conversely, BDS-2 loses to HBD on 2 functions (F4, F11), to BSA-DE on 1 function (F4), and to BSA on 1 functions (F12). We can notice similar results when comparing BDS-1 to BSA, BSA-DE, and HBD, except that BDS-2 performs better on F10 and F12. From these observations, we can conclude that our BSA-DE-SA approach clearly outperforms BSA, and gives slightly better results than BSA-DE and HBD. Figure 1 shows the convergence curves for selected benchmark problems and it highlights that our hybrid approach leads to faster convergence : we can see that BDS-2 saves between 45% and 70% of function evaluations compared to BSA-DE and HBD for F8, about 40% compared to BSA-DE for F9, and between 25% and 45% compared to BSA-DE and HBD for F13. Moreover, BDS-2 is the only algorithm that solves F10 within the allowed function evaluation budget.

4 Contribution to speed up BSA runtime

The graphics processing unit (GPU) has a highly parallel architecture, and it can be easily programmed for general purpose computations with high-level languages, thanks to dedicated parallel computing platforms like CUDA for NVIDIA GPU devices. The CUDA platform allows to realize heterogeneous parallel computations, which means that the program is launched on the CPU, that delegates parallel subroutines (so-called kernels) to the GPU. In CUDA pro-

Table 3. Basic statistics of the two versions of BSA-DE-SA, and comparison with BSA [2], BSA-DE [3], and HBD [7] (Mean: mean error; Std: standard deviation; Best: best error). Best values are depicted in bold font.

ID	Statistics	BDS-1	BDS-2	BSA [2]	BSA-DE [3]	HBD [7]
F1	Mean	**0**	**0**	3.45331725e-1	**0**	4.69223633e-5
	Std	**0**	**0**	3.56207055e-1	**0**	4.87788549e-5
	Best	**0**	**0**	4.65828600e-2	**0**	1.74837295e-6
F2	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	Best	**0**	**0**	**0**	**0**	**0**
F3	Mean	**0**	**0**	3.31653019e-2	**0**	1.65826509e-1
	Std	**0**	**0**	1.81653839e-1	**0**	5.27993560e-1
	Best	**0**	**0**	**0**	**0**	**0**
F4	Mean	9.30325416e-1	1.32887461	2.35616889e+1	**6.64437376e-01**	8.01149354e-1
	Std	1.71491464	1.91143983	2.90306080e+1	**1.51112585**	1.62101635
	Best	**0**	**0**	5.31405876e-7	**0**	**0**
F5	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	Best	**0**	**0**	**0**	**0**	**0**
F6	Mean	**0**	**0**	8.12184166e-7	**0**	**0**
	Std	**0**	**0**	1.18619825e-6	**0**	**0**
	Best	**0**	**0**	**0**	**0**	**0**
F7	Mean	1.88063034e+3	**6.70067111e+2**	1.62772681e+4	6.63797991e+3	5.12822952e+3
	Std	4.09511408e+3	**8.99497851e+2**	2.63103587e+4	5.96963034e+3	6.89120964e+3
	Best	6.85806410	**1.69290665e-1**	3.23132561e+2	1.23221979e+2	1.28697388e+1
F8	Mean	**0**	**0**	3.52038638e-3	**0**	**0**
	Std	**0**	**0**	1.00832481e-2	**0**	1.41395434e-8
	Best	**0**	**0**	1.16021564e-5	**0**	**0**
F9	Mean	**0**	**0**	1.63586845e-2	**0**	5.28382701e-5
	Std	**0**	**0**	3.29592107e-2	**0**	6.56037241e-5
	Best	**0**	**0**	1.06714993e-4	**0**	2.68750955e-6
F10	Mean	1.32885971e-1	**0**	2.31962945e-1	1.32889360e-1	5.79353282e-4
	Std	7.27846435e-1	**0**	5.86248030e-1	7.27845795e-1	3.01367607e-3
	Best	**0**	**0**	**0**	**0**	**0**
F11	Mean	5.42895964e-2	4.61309502e-2	6.56037488e-2	1.14081123e-1	**3.33610373e-2**
	Std	4.71316146e-2	2.29246572e-2	3.49897515e-2	5.14108950e-2	**2.15975637e-2**
	Best	7.52199899e-3	9.85728587e-3	3.43988696e-4	3.66388264e-2	**0**
F12	Mean	2.03415389e+1	2.03230528e+1	**2.03225585e+1**	2.03462701e+1	2.03325172e+1
	Std	**7.02011419e-2**	8.34903645e-2	8.21386118e-2	7.14983620e-2	7.80534782e-2
	Best	2.01888263e+1	**2.00865221e+1**	2.01202686e+1	2.02124186e+1	2.02032472e+1
F13	Mean	**0**	**0**	**0**	**0**	**0**
	Std	**0**	**0**	**0**	**0**	**0**
	Best	**0**	**0**	**0**	**0**	**0**

gramming, each kernel is a piece of code called from the CPU and duplicated on the GPU to be executed in parallel on multiple data (the GPU has a SIMD architecture, i.e. single-instruction multiple-data). Each kernel duplicate is executed by a CUDA thread, and all these threads are organized as follows: each kernel call creates a grid composed of thread groups, called blocks, that all contain the same number of threads. Thus, in order to take advantage of the GPU performance, any evolutionary algorithm should be adapted, in terms of data decomposition, to be processed in parallel by blocks of threads [4–6].

The first feature of our proposed CUDA implementation is that we delegate to the GPU the most time-consuming part of the algorithm, that is the evaluation of the population. This can be done with two levels of parallelization as follows. Firstly, the evaluations of all individuals can be done in parallel. And secondly, since for the most part of the benchmark functions we need to perform the same computations on each dimension before aggregating the results (for example, with a sum), the dimensions can also be processed in parallel. Getting back to CUDA programming, it means that the evaluation workload can be divided into N blocks of D threads, that each deals with 1 dimension of 1 individual.

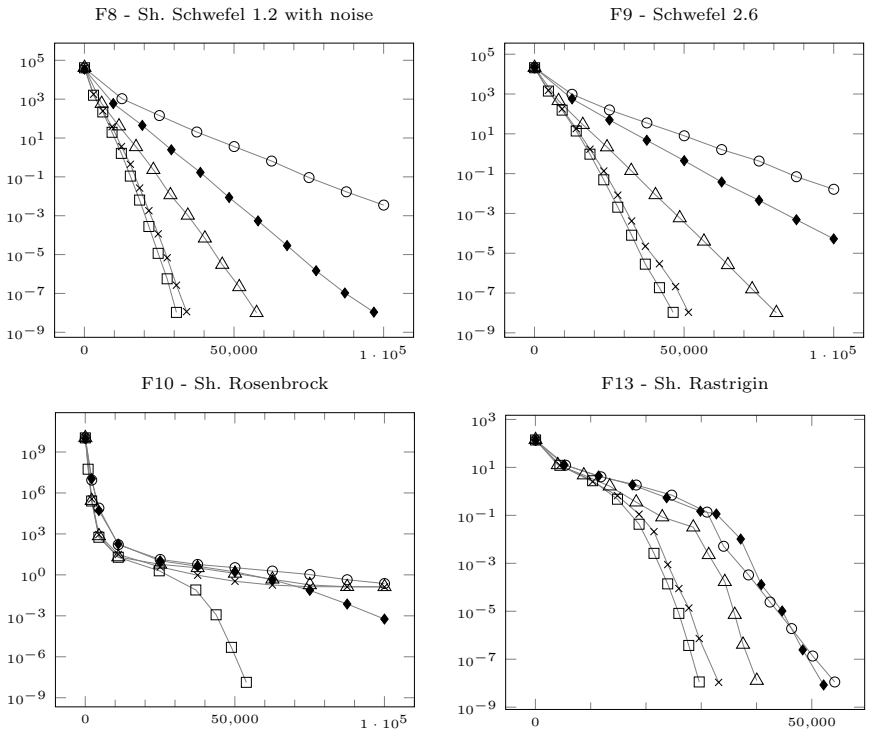


Fig. 1. The curves show how many function evaluations (x-axis) are needed to reach a certain mean error (y-axis in log scale) for selected benchmark problems of Table 1. BSA is depicted with empty circles, BSA-DE with empty triangles, HBD with filled diamonds, BDS-1 with crosses, and BDS-2 with empty squares.

However, as already noticed in the literature [6], if the evaluation is the only task entrusted to the GPU, the algorithm has to transfer the whole population from CPU memory to GPU in every generation, which is very slow compared to arithmetic computations on GPU. Therefore, we choose to store the population in the GPU global memory in order to minimize the time lost in data transfer. It means that all steps of the algorithm are processed by the GPU, while the generation loop is done by the CPU, that launches a GPU kernel for each step with the ad-hoc data decomposition, in terms of CUDA blocks and threads. As much as possible, we divide the processings into N blocks of D threads: as seen above, this is particularly suited to evaluate the population, but also, for example, to generate the initial population, to apply the mutation equation, or to perform the boundary control. In addition to that, other decompositions are sometimes needed, depending on the processing to be realized: for example, 1 block of N threads to find the best individual, or 1 block of D threads to update the global best solution.

Table 4. Comparison of BSA, BDS-1, and BDS-2 in high dimensions (Mean: mean solution; Time: mean runtime in seconds). Best values are depicted in bold font.

N=D	ID	Statistics	BSA [2]	BDS-1			BDS-2		
			CPU	CPU	GPU	Speedup	CPU	GPU	Speedup
128	F1	Mean	3,2531e+3	2,3844e+3	2,6854e+3		**1,9063e+3**	2,0242e+3	
		Time	11,13	11,25	**2,89**	3,90	11,53	19,49	0,59
	F2	Mean	**4,5019e-2**	2,6885	2,7312		2,4161	2,5679	
		Time	3,41	3,61	**2,97**	1,22	3,71	19,45	0,19
	F3	Mean	1,6949e+2	1,1462e+2	**1,1045e+2**		1,2092e+2	1,2230e+2	
		Time	3,80	3,88	**2,64**	1,47	4,07	19,58	0,21
	F4	Mean	5,2074e+2	3,5942e+2	3,2152e+2		2,7981e+2	**2,5426e+2**	
		Time	**2,87**	3,05	3,03	1,01	3,23	19,50	0,17
	F5	Mean	**1,2849**	1,1354e+1	1,1646e+1		1,1436e+1	1,1518e+1	
		Time	63,67	64,41	**3,71**	17,37	64,80	20,39	3,18
	F14	Mean	-9,3241e+1	**-9,8617e+1**	-9,8456e+1		-9,8065e+1	-9,8098e+1	
		Time	9,34	9,45	**2,74**	3,45	9,65	19,65	0,49
256	F1	Mean	**7,4732e+3**	1,5148e+4	1,5590e+4		1,4160e+4	1,4756e+4	
		Time	80,87	81,20	**6,49**	12,51	82,47	73,08	1,13
	F2	Mean	**1,1330**	4,4814	4,6626		4,4919	4,2836	
		Time	14,96	15,04	**6,35**	2,37	15,65	72,97	0,21
	F3	Mean	6,2993e+2	6,0159e+2	**5,9217e+2**		6,1518e+2	6,2133e+2	
		Time	15,23	15,63	**5,63**	2,78	16,33	73,55	0,22
	F4	Mean	2,0790e+3	1,2885e+3	1,3246e+3		**8,8510e+2**	9,3211e+2	
		Time	11,38	12,02	**6,58**	1,83	12,70	73,09	0,17
	F5	Mean	**1,3822e+1**	7,6411e+1	7,6420e+1		7,7453e+1	7,7002e+1	
		Time	256,59	262,90	**8,65**	30,40	263,94	75,39	3,50
	F14	Mean	-1,4556e+2	**-1,5584e+2**	-1,5502e+2		-1,5358e+2	-1,5336e+2	
		Time	37,74	37,57	**5,97**	6,29	38,42	73,97	0,52
512	F1	Mean	**1,2895e+4**	6,0397e+4	6,6561e+4		5,8543e+4	5,9229e+4	
		Time	613,77	614,39	**20,46**	30,02	620,01	289,71	2,14
	F2	Mean	**2,7341**	7,2895	7,0967		6,9065	6,9606	
		Time	61,69	61,76	**18,05**	3,42	63,96	286,98	0,22
	F3	Mean	**1,8488e+3**	2,0890e+3	1,9805e+3		2,1301e+3	1,8501e+3	
		Time	60,65	63,28	**15,60**	4,06	66,01	286,14	0,23
	F4	Mean	8,4335e+3	1,6159e+4	1,3098e+4		**5,9431e+3**	6,0513e+3	
		Time	45,65	47,92	**18,31**	2,62	50,77	287,76	0,18
	F5	Mean	**6,1091e+1**	2,6234e+2	2,6643e+2		2,6058e+2	2,5402e+2	
		Time	1027,70	1062,34	**26,48**	40,12	1066,46	295,91	3,60
	F14	Mean	-2,1917e+2	-2,3382e+2	**-2,3394e+2**		-2,3102e+2	-2,3067e+2	
		Time	151,40	151,35	**16,87**	8,97	155,07	290,21	0,53

We realized an experimental study in order to compare our GPU implementations of BDS-1 and BDS-2 with sequential BSA [2]. For reasons of dimensional scalability, these algorithms have been tested on the benchmark functions F1-5 of Table 1 and on Michalewics function (denoted as F14, and defined on $[0, 3.1416]$, according to [2]). The control parameters of each algorithm have been set as shown in Table 2, except the population size that now depends on the problem dimension as follows: $N = D$. Several experiments have been conducted with $D = 128$, $D = 256$, and $D = 512$. For a given value of D, each algorithm has been run 15 times on each benchmark problem, and $3\,000 \times D$ function evaluations per run were allowed. For these experimentations, all the compared algorithms are written in C/C++, and the corresponding programs are compiled on an Intel Core processor i5-3330 CPU (3.00GHz) with 4 GB of RAM and a NVIDIA GeForce GTX680 GPU.

Table 4 reports basic statistics for the compared algorithms. First of all, it seems that BSA finds solutions of better quality than BDS-1 and BDS-2. However, all compared results almost always have the same order of magnitude. We can also see that BDS-1 ties with BDS-2 in terms of solution quality: roughly, BDS-1 is generally better for F3 and F14, whereas BDS-2 tends to win for F1, F2 and F4. Secondly, the resulting mean runtimes show that BDS-1 GPU version can

lead up to a 40 time speedup with regard to BDS-1 CPU version. It sounds that the acceleration mainly comes from the evaluation of the population, and that it directly depends on the computation complexity of the considered benchmark function. Thirdly, we can notice that BDS-2 speedup is much lower than that of BDS-1. It is due to the HBD part of BDS-2: one level of parallelization is lost in this part of the GPU algorithm, since Section 2.2 and Table 2 point out that all HBD evolutionary operators are applied only for a few individuals ($N/30$). So, almost all the speedup gained from BSA iteration is then lost in the DE iteration needed for the HBD part of BDS-2. In a word, we can conclude that BDS-1 GPU version seems to be the most suitable for the selected high dimensional benchmark problems.

5 Conclusion

A hybrid BSA-DE-SA algorithm has been presented and an experimental study on 13 benchmark problems shows that it performs well in terms of solution quality and convergence speed. Then, the design of our GPU implementation has been explained, and experimental results point out that a significant speedup can be achieved, up to 40 times with regard to sequential program.

In future work, we will consider comparing our approach to other algorithms (for example, PSO, CMAES, SHADE) with additional benchmark functions. As we introduce new user-defined parameters, another perspective would be to improve the proposed algorithm with a self-adaptive technique, in order to be less user-dependent and to achieve possibly better results. Finally, in the longer term, it would be interesting to compare this hybridization with existing large-scale optimization methods.

References

1. M. Brévilliers, O. Abdelkafi, J. Lepagnot, and L. Idoumghar. Idol-guided backtracking search optimization algorithm. In *12th International Conference on Artificial Evolution - EA 2015, Lyon, France*, October 2015.
2. P. Civicioglu. Backtracking search optimization algorithm for numerical optimization problems. *Applied Mathematics and Computation*, 219(15):8121 – 8144, 2013.
3. S. Das, D. Mandal, R. Kar, and S. Prasad Ghoshal. A new hybridized backtracking search optimization algorithm with differential evolution for sidelobe suppression of uniformly excited concentric circular antenna arrays. *International Journal of RF and Microwave Computer-Aided Engineering*, 25(3):262–268, 2015.
4. V. Kalivarapu and E. Winer. A study of graphics hardware accelerated particle swarm optimization with digital pheromones. *Structural and Multidisciplinary Optimization*, 51(6):1281–1304, 2015.
5. G.-H. Luo, S.-K. Huang, Y.-S. Chang, and S.-M. Yuan. A parallel bees algorithm implementation on GPU. *Journal of Systems Architecture*, 60(3):271 – 279, 2014.
6. P. Pospichal, J. Jaros, and J. Schwarz. Parallel genetic algorithm on the CUDA architecture. In *Applications of Evolutionary Computation: EvoApplications 2010*, pages 442–451. Springer Berlin Heidelberg, 2010.
7. L. Wang, Y. Zhong, Y. Yin, W. Zhao, B. Wang, and Y. Xu. A hybrid backtracking search optimization algorithm with differential evolution. *Mathematical Problems in Engineering*, 2015.

A New Parallel Memetic Algorithm to Knowledge Discovery in Data Mining

Dahmri Oualid[1,*], Ahmed Riadh Baba-Ali[2]

[1] Computer Science Department, FEI, USTHB,
BP 32 El Alia, BabEzzouar Algeria
dahmri_oualid_39@yahoo.fr
[2] Research Laboratory LRPE, FEI, USTHB,
BP 32 El Alia, BabEzzouar Algeria
riadhbabaali@yahoo.fr

Abstract. This paper presents a new parallel memetic algorithm (PMA) for solving the problem of classification in the process of Data Mining. We focus our interest on accelerating the PMA. In most parallel algorithms, the tasks performed by different processors need access to shared data, this creates a need for communication, which in turn slows the performance of the PMA. In this work, we will present the design of our PMA, In which we will use a new replacement approach, which is a hybrid approach that uses both Lamarckian and Baldwinian approaches at the same time, to reduce the quantity of informations exchanged between processors and consequently to improve the speedup of the PMA. An extensive experimental study performed on the UCI Benchmarks proves the efficiency of our PMA. Also, we present the speedup analysis of the PMA.

Keywords: parallel memetic algorithm, classification, extraction of rules, Lamarckian approach, Baldwinian approach, hybridization.

1 Introduction

Nowadays there is a huge amount of data being collected and stored in databases everywhere across the globe, and there are invaluable informations and knowledge "hidden" in such databases, and without automatic methods for extracting this informations, it is practically impossible to use them.
Data mining [1], was born for this need. Among the tasks of this process, we find the supervised classification [2] is one of the most important. It consists of predicting a certain outcome based on a given input. In order to predict the outcome, the algorithm processes a training set containing a set of attributes and the respective outcome, usually called goal or prediction attribute. The algorithm tries to discover relationships between the attributes that would make it possible to predict the outcome. Next, the algorithm is given a data set not seen before, called prediction set, which contains the same set of attributes, except for the prediction attribute – not yet known. The algorithm analyses the input and produces a prediction. The prediction accuracy defines

how “good” the algorithm is. This problem is NP-hard [3] and for that reason an exponential complexity making impossible the use of exact methods when the data size is large.
Meta-heuristics [4] [5] are algorithms that can provide a satisfactory solution in a relatively short time on this class of problems. Among these methods, we are particularly interested in the Memetic Algorithms[18] (hybridization of a local search [7] and genetic algorithm [6]). The genetic algorithm is so widely used to solve data mining classification problems is the fact that prediction rules are very naturally represented in GA. Additionally, GA has proven to produce good results with global search problems like classification. But this kind of algorithms requires considerable computation time and amount of memory which are closely related to the size of the problem and to the quality of the solution to obtain.
Therefore, these algorithms become interesting to parallelize. In general, parallelism is used to solve complex problems requiring expensive algorithms in terms of execution time. But in most parallel algorithms, the tasks performed by different processors need access to shared data, this creates a need for communication which in turn slows the performance of the parallel algorithm. These communications are even more influential, in the case where processors require data generated by other processors. So the objective of this work is to minimize communications in terms of data volume and frequency of exchanges without penalizing the quality of the solution.

2 Related work

Genetic Algorithms are those among which have been the subject of the greatest number of parallelization work, particularly because of their fundamental parallel nature [8]. Cantú-Paz [9] presented a review of the main publications related to parallel genetic algorithms. They distinguish three main categories of parallel genetic algorithms :

- Parallelization form master-slave on a single population
- Parallelization Fine-grained on a single population (diffusion model)
- Parallelization Coarse-grained on multiple populations (migration model)

In the first model, there is only one population residing on a single processor called the master. This one makes the different genetic operators of the algorithm on population and then distributes the evaluation of individuals to slave processors.
In the second model, which is suitable for massively parallel computers, the individuals in the population are distributed on processors, preferably at a rate of one individual per processor. Selection and reproduction of individuals operators are limited to their respective neighborhoods. However, as the neighborhoods overlap (an individual may be part of the vicinity of several other individuals), a certain degree of interaction between all individuals is possible.
The third category, more sophisticated and more popular, consists of several populations that are distributed over processors. These can evolve independently of each other with only occasional exchanges of individuals. This optional exchange called the migration phenomenon, is controlled by various parameters and generally pro-

vides a better performance of this algorithm type. This category is also called "parallel genetic algorithms islands".

2.1 Hybrid parallelization of metaheuristics

Each metaheuristic has its own characteristics and its own way to look for solutions. Therefore, it may be interesting to hybridize several different metaheuristics to create new research behaviors. In this regard, Bachelet et al. [10] identified three main forms of hybrid algorithms:

- Sequential hybrid, where two algorithms are executed one after the other, the results provided by the first being the initial solutions of the second.
- Synchronous parallel hybrid, where a search algorithm is used in place of an operator. An example of this type is to replace the mutation operator of genetic algorithm with a tabu search.
- Asynchronous parallel hybrid, where several search algorithms work concurrently and exchange informations.

2.2 Measuring Performance of parallel algorithms

In general, it's hard to make fair comparisons between algorithms such as metaheuristics. The reason is that we can infer different conclusions from the same results depending on the metrics we use and how they are applied. This comparison become more complex when compared parallel metaheuristics, it's way is necessary to qualify some metrics, or even to adjust them to better compare parallel metaheuristics between them. Alba et al. [11] indicate that for non-deterministic algorithms, such as meta-heuristics, it is the average time of sequential and parallel versions which must be taken into account. It offers different definitions of speedup. Strong speedup which compares the parallel algorithm with the result of the best known sequential algorithm. This is what is closest to the true definition of speedup but considering the difficulty of finding each time the best existing algorithm, this standard is not used much. Speedup is called weak if we compare the parallel algorithm with the sequential version developed by the same researcher. It can then present its progress both in terms of quality and in pure speedup. Barr and Hickman [12] presented a different taxonomy consisting of relative speedup and absolute speedup. The relative speedup is the ratio between the parallel version running on a single processor and that performed on the set of processors. Finally, the absolute speedup, which is the ratio of the fastest sequential version on any machine and the execution time of the parallel version.

Speedup. The first and probably most important performance measure of a parallel algorithm is the speedup [11]. It is the ratio of the execution time of the best algorithm known on 1 processor and that of the parallel version. Its general formula is:

$$Speedup = \frac{\text{Sequential execution time}}{Parallel\ execution\ time}$$

Efficiency. Another popular metric is efficiency. It gives an indication of the rate of use of the requested processors. Its value is comprised between 0 and 1 and it can be expressed as a percentage. The more the value of efficiency is close to 1, the better is the performance. Efficiency equal to 1 matches to a linear speedup. Its general formula is :

$$Efficiency = \frac{Speedup}{\mathrm{P}}$$

(***P*** is the number of processors)

Other measures. Among other metrics used to measure the performance of parallel algorithms, we find the "scaled speedup" (expandable speedup) [11] which measures the use of available memory. We also find the "scaleup" (scalability) [11] to measure the ability of the program to increase its performance when the number of processors increases.

2.3 Impact of communication on the performance of parallel algorithms

The measure of parallel performance is a complex metric. This is mainly due to the fact that the parallel performance factors are dynamic and distributed. [13] The communication factor is among the most influential on the performance of the algorithm. In many parallel programs, the tasks performed by different processors need access to shared data. This creates a need for communication and slows the performance of the algorithm. These communications are more important in the case where processors require data generated by other processors. These communications are minimized in terms of data volume and frequency of exchanges when we used our new replacement approach, which is a hybrid approach that uses both Lamarckian and Baldwinian approaches at the same time, and this is the object of the next section.

2.4 Lamarckianism vs. Baldwinian effect

When integrating local search with genetic algorithm we are faced with the dilemma of what to do with the improved solution that is produced by the local search. That is, suppose that individual i belongs to the population P in generation t and that the fitness of i is f(i). Furthermore, suppose that the local search produces a new individual i' with f(i') < f(i) for a minimisation problem. The designer of the algorithm must now choose between two alternative options. Either (option 1) he replaces i with i', in which case P = P −{i}+{i'} and the genetic information in i is lost and replaced with that of i', or (option 2) the genetic information of i is kept but its fitness altered : f(i)= f(i'). The first option is commonly known as Lamarckian learning while the second option is referred to as Baldwinian learning (Baldwin, 1896). The issue of whether natural evolution was Lamarckian or Baldwinian was hotly debated in the nineteenth century until Baldwin suggested a very plausible mechanism whereby evolutionary progress can be guided towards favorable adaptation without the inheritance of lifetime acquired features. Unlike in natural systems, the designer of a Memetic Algorithm may want to use either of these adaptation mechanisms. Hinton and Nowlan (1987) showed that the Baldwin effect could be used to improve the evolution of arti-

ficial neural networks, and a number of researchers have studied the relative benefits of Baldwinian versus Lamarckian algorithms, e.g., Whitley et.al. (1994), Mayaley (1996), Turney (1996), Houck et.al. (1997), etc. Most recent work, however, favored either a fully Lamarckian approach, or a stochastic combination of the two methods. It is a priori difficult to decide what method is best, and probably no one is better in all cases. Lamarckianism tends to substantially accelerate the evolutionary process with the caveat that it often results in premature convergence. On the other hand, Baldwinian learning is more unlikely to bring a diversity crisis within the population but it tends to be much slower than Lamarckianism.
In our PMA, in each slave machine, when the Tabu Search algorithm runs on individuals sent by the master machine, and before returning improved individuals, we have to decide which replacement strategies will be applied. This decision will be taken according to the fitness value of the improved individual. When this fitness is lower than predefined threshold, we don't need to the genetic information of the individual, but we have to send his fitness to the master, in this case, we will send just the fitness value of the individual without its genetic information to the master to replace it in population with the Baldwinian approach, otherwise if the fitness value of the improved individual is above then the predefined threshold , in this case, we need to send the genetic information and the fitness value of the individual to the master to replace it in population with the Lamarckian approach.

3 Adaptive Memetic Algorithm

We present the adaptation of the Memetic Algorithm (MA) [14],[15] for the Classification problem. In the literature, there are two different approaches to extract rules using a genetic algorithm: the Pittsburgh approach and the Michigan one [15]. In our work we have chosen the Michigan approach where a classification rule presents the following form :

$$A \longrightarrow C$$

A is the premise or antecedent of the rule and C the predicted class. The A part of the rule is a conjunction of terms that are of the form :

Attribue	Operator	Value

The rule coding involves a sequence of genes arranged in the same order as the attributes of the studied data except for the last gene of the individual or chromosome which contains the predicted value of class [16]. Each condition is coded by a genome and consists of a triplet of the form (Ai op Vij), where Ai is the ith table attribute on which the algorithm is applied. The term op is one of the operators '=', '<' or '>' and Vij is the Ai attribute value belonging to its values domain. To each genome is associated a boolean field that indicates whether the premise is activated or not, in order to maintain the chromosome size fixed. Even if individuals have the same length, the rules associated with them are of variable length. The structure of an individual is shown in Figure 1, where m is the total number of attributes.

Fig.1. Structure of an individual

The initial population is randomly generated to give it some diversity. Each individual (or rule) is a potential solution to the problem to solve. However, these solution do not all have same relevance degree. The rule coding involves a sequence of genes arranged in the same order as the attributes of the studied data except for the last gene of the individual of chromosome which contains the predicted value of the class. This is why the following criteria have been chosen [16] :

- To maximize the rule converge;
- To maximize the accuracy rate of the rule;
- To minimize the rule size because the comprehensibility of the rule is measured by the number of premises;

Fitness =

Λ_1 * Coverage / Total number of instance
+ Λ_2 * TP / Coverage
- Λ_3 * Rule size / Total number of attributes

where Λ_i is a real value that verifies $\sum \Lambda_i = 1$

In our Memetic Algorithm, we used hybridization of the tabu search with a genetic algorithm. we used the tournament selection and the classical genetic crossover and mutation operators. The individual resulting from crossover and mutation operators is the initial solution (a rule) for the tabu search, then the best individual found by the tabu search will replace the worst individual in term of accuracy in the population of the genetic algorithm and so on.
In the tabu search approach, the neighborhood of the initial solution consists of all solutions obtained by performing a one-movement operator which is applied to the current individual as many times as the number of attributes of the considered training set. So the created neighbors are evaluated by computing the same fitness as in the genetic algorithm. Then the best solution in the vicinity of the current individual is added to tabu list. Thus, the worst individual in term of accuracy is destroyed if the size of tabu list is exceeded and so on.

4 The proposed PMA architecture

We present in this section the design of our synchronous parallel Memetic Algorithm (PMA). It is a synchronous parallel model based on master-slave form uses a unique population residing on a single processor called the master. The latter performs the different genetic operations of the algorithm and then distributes the Tabu Search on the slave processors.

4.1 Replacement strategy used

In our PMA we hybridized the Lamarckian and Baldwinian approaches together to create a new approach in order to reduce the genetic information exchanged between the Genetic algorithm and the Tabu Search algorithm without penalizing the accuracy of the classifier based on our PMA. This hybrid approach is defined as follows:

- If the local search produces an individual i' with f(i')>Threshold, in this case the Lamarckian approach is used, therefore P = P - (i) + (i') and f(i) = f(i')
- If the local search produces an individual i' with f(i')<=Threshold, in this case, the Baldwinian approach is used, therefore, P still the same and f(i) = f(i')

The Threshold is a variable parameter, its value determines the number of individuals which will be replaced with the Lamarckian approach, and the number of individuals that will be replaced with the Baldwinian approach.

4.2 Our synchronous PMA using Master-Slave Model

In this model, we have a master machine and the others are slave machines. In each slave machine, the Tabu Search algorithm runs on individuals sent by the master machine and before returning improved individuals we compare the fitness value of each individual to a predefined threshold. If the fitness value of the individual is more than the threshold then the genetic information and the fitness value of the individual are both sent to the master, otherwise, if the fitness value of the individual is lower than the threshold, we will send just the fitness value of the individual without its genetic information.
The memetic algorithm runs in the master processor, and the master is the only machine that has the overall population in its own memory. The master processor performs the selection, the crossover and the mutation of individuals and then distribute them to the slaves. Each slave processor receives the individuals, performs the tabu search and returns the optimized individuals to the master. When the master processor receives all results from slaves, he performed the replacement operation. If the master processor receives the fitness value of the an improved individual with his genetic information, then he replaced it in a population with the Lamarckian approach, else if he receives the fitness value of the improved individual without his genetic information, then he replaced it in a population with the Baldwinian approach.
The learning database is the only common data between the master and slaves. Consequently, we find the same learning database in all slaves. In order to always have the same learning database anywhere, the master machine sends the best individual selected after each generation of the Memetic Algorithm to all slave machines, for that they can update their learning database.

Master/Slave communication. The different types of communication can be summarized as follows :

From master to slave. The different informations sent from the master to a slave are:

- The individual resulting from the selection, crossover and mutation operators;
- The threshold value after each iteration;
- The best individual of each generation;

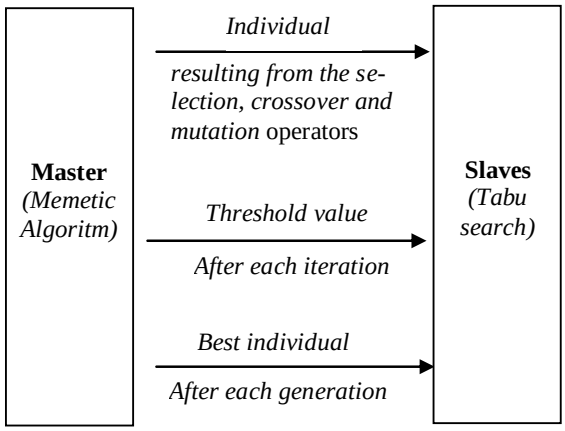


Fig.2. Communication from master to slave

From slave to master. The different informations sent from a slave to the master are :

- The fitness value of the improved individual with its genetic information.
- The fitness value of the improved individual without its genetic information.

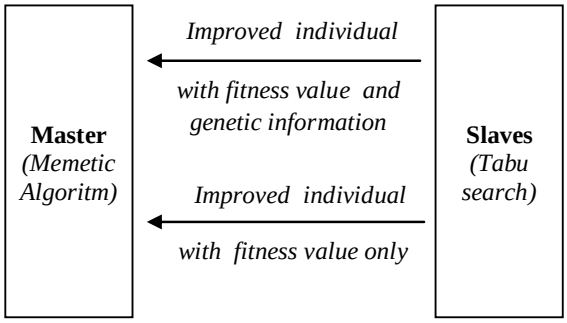
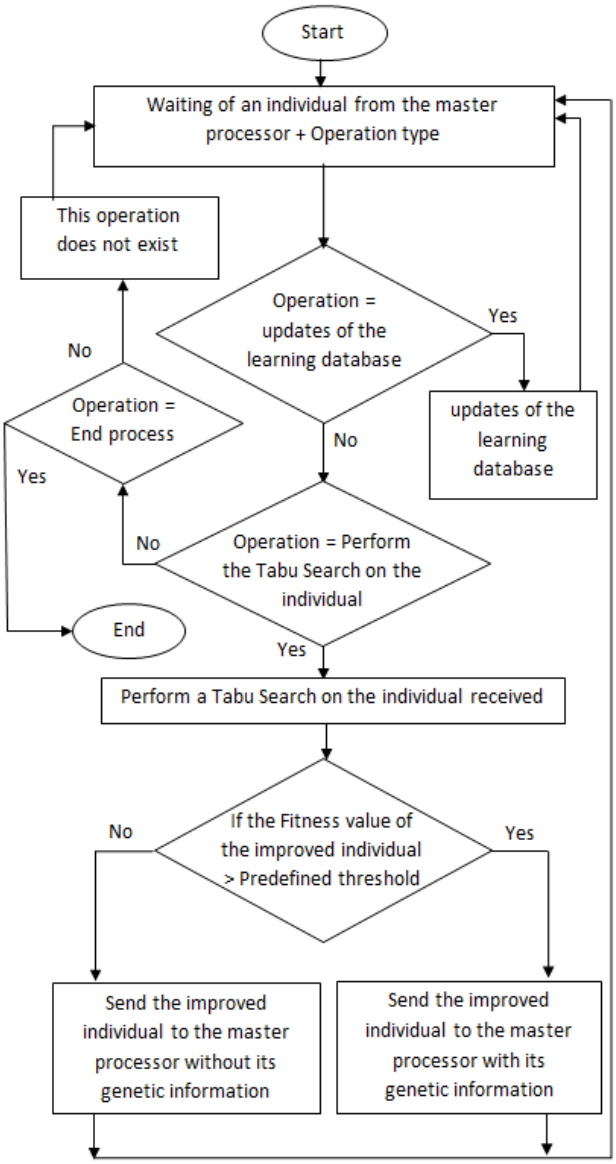


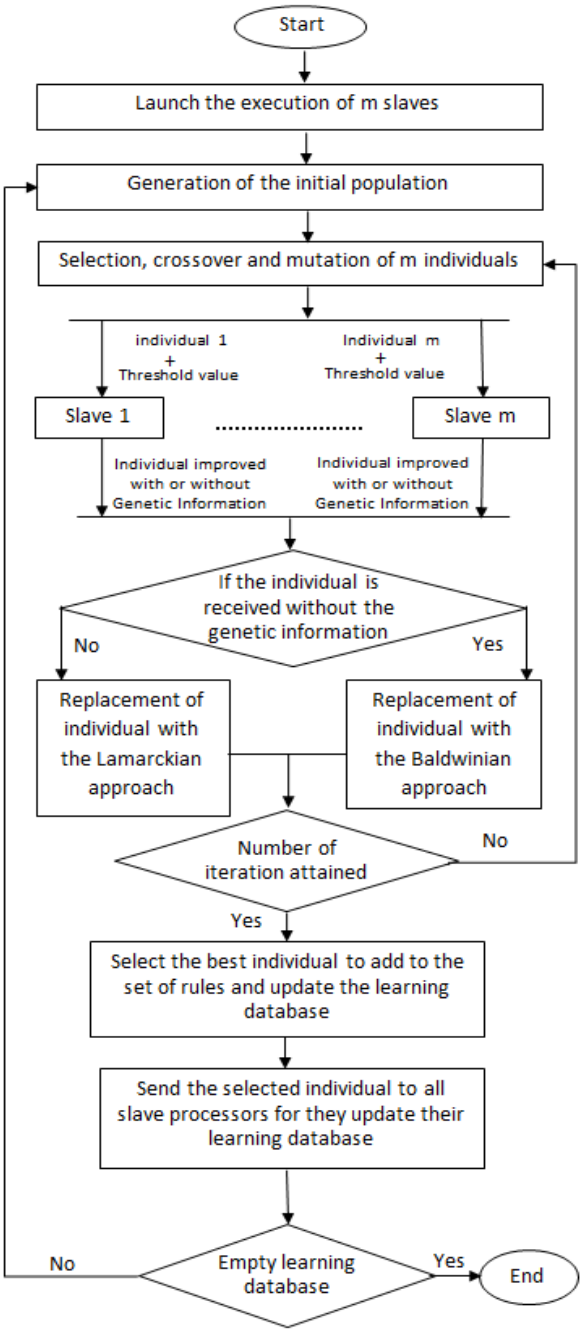
Fig.3. Communication from slave to master

Synchronization. In this model the synchronization is launching of different slave processors. At first, the master launches them all, then each time the master needs to perform the Tabu Search on a set of the individuals distributes them on slave processors and waits for all results, then it replaces them in the population.

Slaves algorithm.



Master algorithm.



5 Results

5.1 UCI Benchmarks

The UCI is a very large database library of Benchmarks selected by the University of California Irvine (UCI)[17]. The latter was made available to the research community in Data Mining. These benchmarks that are widely used and considered as a reference. Hence the importance of using them to evaluate algorithms and compare their performance with other algorithms. Table 1 gives a summary of the databases used in our tests.

Table 1. Databases used

DataBases	Instances number	Attributes number	Class number
hepatitis	155	20	2
heart-statlog	270	14	2
segment-challenge	1500	20	7
ionosphere	351	35	2
kdd-train	11419	42	2
diabetes	768	9	2

5.2 Results obtained by our synchronous PMA

The efficiency of our synchronous PMA is determined by the threshold value. In order to find the best threshold value, we conducted a series of experiments with three different thresholds :

- ***Threshold1:*** Is equal to the worst fitness value in the population for each iteration;
- ***Threshold2:*** Is equal to the best fitness value in the population for each iteration;
- ***Threshold3:*** Is equal to the average of all fitness values of the population for each iteration;

We run the classifier based on our synchronous PMA 10 times successively on the UCI benchmarks given above, for each threshold. We give every time the accuracy obtained, and the percentage of individuals returned without the genetic information compared to individuals returned with the genetic information.
The parameters PC and PM of Memetic Algorithm are PC = 0.025 and PM = 0.8 and the parameters $\lambda 1$, $\lambda 2$ and $\lambda 3$ of the objective function of classifier are $\lambda 1 = 0.1$, $\lambda 2 = 0.8$ and $\lambda 3 = 0.1$ and the parameters memory size and the number of iterations of the Tabu search are 5 and 300.
We have regrouped the average accuracy and the average percentage of individuals exchanged without their genetic information, obtained from all databases for each threshold in the following tables:

Table 2. Averages accuracies

database	Average accuracy (%)		
	Threshold 1	Threshold 2	Threshold 3
hepatitis	84,57	73,30	84,39
heart-statlog	83,79	72,57	83,62
segment-challenge	94,81	83,39	96,06
ionosphere	90,35	87,24	90,14
kdd-train	99,29	86,62	99,79
diabetes	81,46	70,57	81,31

Table 3. Percentage of individuals returned without their genetic information

database	Average percentage of individuals returned without their genetic information (%)		
	Threshold 1	Threshold 2	Threshold 2
hepatitis	0,28	99,84	41,98
heart-statlog	0,16	99,90	40,42
segment-challenge	0.08	99,42	42,36
ionosphere	0,28	99,74	44,58
kdd-train	0,36	99,84	42,48
diabetes	0,06	99,62	43,90

We observe from Tables II and III, that the percentage of individuals returned without their genetic information for the threshold1 is between 0.08% and 0.36% maximum, so most individuals are returned with their genetic information and are replaced with the Lamarckian approach in the population. So with the threshold1, our hybrid approach converges to the Lamarckian approach and we could not reduce the genetic information exchanged between master and slaves. On the other hand, the percentage of individuals returned without their genetic information for the threshold2 is between 99.42% and 99.90%, so most individuals are returned without their genetic information and are replaced with the Baldwinian approach in the population. So with the threshold2, our hybrid approach converges to the Baldwinian approach and we reduced by 50% the genetic information exchanged between the master and his slaves, but on the other hand we obtained bad results in the accuracy of the classifier, for this threshold. For the threshold3 the percentage of individuals returned without their genetic information is between 40.42% and 44.58%, so almost half of the individuals are returned without their genetic information, and also we have obtained very good results in terms of accuracy of the classifier. So with the PMA based on the threshold3, we could decrease by 20% the genetic information exchanged between the master and his slaves without penalizing the accuracy of the classifier.

Furthermore to evaluate the performance of our PMA designed, we have performed a series of tests on a network of 10 computers. The speedup, defined as the quotient

between the time Ts to run the sequential algorithm and the time Tp for the parallel version, is used as the performance criterion.
To test the speedup of our PMA based on the threshold 3, we'll run it on the two databases hepatitis (20 attributes) and kdd-train (42 attributes) and each time we increase the number of slave processors, then we will compare it with the results of another simple PMA(is PMA without the new approach of replacement). The results found are in the following table:

Table 4. Results obtained with a different number of slaves

Number of slave processors	Speedup			
	hepatitis		kdd-train	
	Simple PMA	PMA with threshold3	Simple PMA	PMA with threshold3
1	1,00	1,00	1,00	1,00
2	1,84	1,93	1,65	1,73
3	2,85	2,99	2,56	2,69
4	4,17	4,37	3,75	3,94
5	5,18	5,69	4,14	4,76
6	5,47	6,01	4,37	5,03
7	6,06	6,66	4,52	5,42
8	6,46	7,42	4,68	5,81
9	6,69	7,69	4,80	5,95
10	6,77	7,78	4,84	6,12

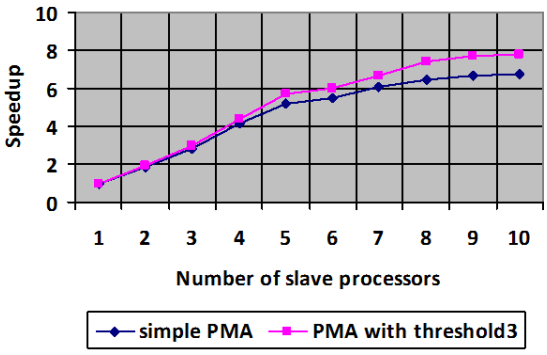


Fig.2 Speedup of the two algorithms for hepatitis database

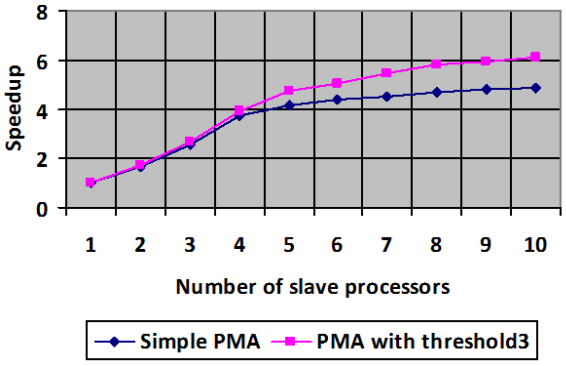


Fig.3 Speedup of the two algorithms for kdd-train database

From the viewpoint of speedup, we observe from Table IV and both Figures 1 and 2 that the simple PMA and the PMA with threshold3 give both good results, every time we increase the number of slave processors, the speedup also increases. But if we compare the speedup of the two algorithms, we observe that they have almost the same speedup when the number of slaves is between 1 and 4, but once the number of slaves exceeds 4 the speedup of PMA with threshold3 becomes better than that of simple PMA for both databases hepatitis and kdd-train, which can be justified by the increase in the cost of communication between the slaves and the master for the simple PMA, the fact that the number of slaves increases the size of exchanged messages also increases, therefore, the communication costs slow the speedup. On the other hand the speedup of the PMA with threshold3 is better because the size of the messages exchanged is reduced by 20%, therefore, the cost of communications is reduced too, and the speedup is increased.

We also observe that the speedup of the two algorithms for hepatitis database is better than their speedup for kdd-train database, which can be justified by the number of attributes of the two databases. The fact that the number of attributes of kdd-train database (42 attributes) is twice the number of attributes of hepatitis database (20 attributes), the size of exchanged messages and the cost of communications is also double.

6 Conclusion

In this work, we presented the design of our parallel Memetic Algorithm for building a classifier. In which we used a new replacement approach, which is a hybrid approach that uses both Lamarckian and Baldwinian approaches at the same time, to reduce the quantity of information exchanged between the master and his slaves.

In order to see the effectiveness of this new hybrid approach of replacement and their effect on the quantity of information exchanged and on the accuracy of the classifier, we performed a series of tests on the UCI Benchmarks, and through the tests, it was found that we have decreased by 20% the quantity of information exchanged between the master and his slaves without penalizing the accuracy of the classifier.

To show the performance of our parallel Memetic Algorithm, we performed a series of tests on a network of 10 computers. Then we compared the speedup obtained by our parallel Memetic Algorithm with the speedup of another simple parallel Memetic Algorithm. It was observed that once the number of slaves exceeds 4, the speedup of our parallel algorithm is better than the simple parallel algorithm, because the number of messages exchanged in our parallel algorithm decreased by 20%, therefore the communication costs are reduced and the speedup is increased. It was also noted that if the number of attributes in the database used increases, therefore, the size of exchanged messages and the cost of communications also increases, hence the importance of our work in minimizing the quantity of exchanged individuals.

References

1. K. J. Cios, W. Pedryecz, R. W. Swinniarsky et L. A. Kurgan. « Data Mining : A Knowledge Discovery Approach .» Editions Springer Science. (2007).
2. A. K. Jain et R. C. Dubes. « Algorithms for clustering data.» Editions Prentice Hall Advanced Reference Series. (1988).
3. Johann Dréo, Alain Pétrowski, Patrick Siarry, Eric Taillard. « Métaheuristiques pour l'optimisation difficile ». Eyrolles, (2005).
4. Christian Blum, Andrea Roli. « Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison » . ACM Computing Survey, Vol. 35 No 3, (Sept. 2003).
5. Jin-Kao Hao, Philippe Galinier, Michel Habib. « Métaheuristiques pour l'optimisation combinatoire et l'affectation sous contraintes ». Revue d'intelligence artificielle, (1999).
6. Goldberg D.E., « Genetic Algorithms in Search, Optimization and Machine Learning ». Addison Wesley, Massachusetts, (1989).
7. Glover, F. "Tabu Search - Part I," ORSA Journal on Computing, 1(3), 190-206. Glover, F. (1989).
8. Crainic, T. G. et Toulouse, M. (1998). Parallel Metaheuristics. Fleet Management and Logistics. T. G. C. a. G. Laporte. Norwell, MA., Kluwer Academic: 205-251.
9. Cantú-Paz, E. (1998). "A survey of parallel genetic algorithms." Calculateurs parallèles, réseaux et systèmes répartis 10(2): 141-171.
10. Bachelet, V., Hafidi, Z., Preux, P. et Talbi, E.-G. (1998). "Vers la coopération des métaheuristiques." Calculateurs parallèles, réseaux et systèmes répartis 10(2).
11. ALBA, E., AND LUQUE, G. IV leasuring the performance of parallel metaheuristics. In Parallel Metaheuristics : A new Class of Algorithms. Wiley-Interscience, 2005.
12. BARR, R., AND HICKMAN, B. Reporting Computational Experiments with ParaUel Algorithms : Issues, Measures, and Experts' Opinions. Dept. of Computer Science and Engineering, Southern Tvlethodist University, 1992.
13. MALONY, A. Tools for parallel Computing : A Performance Evaluation Perspective. Springer, 2000, ch. VII, p. 342.
14. Bacardit, J.: Pittsburgh Genetic-Based Machine Learning in the Data Mining era: Representations, Generalization, and Run-time. Phd Thesis, Universitat Ramon LIul, Spain (2004)
15. Witten, I.H.: Data Mining: Practical Machine Learning Tools and Techniques with JAVA Implementations. Morgan Kaufman Publishers, San Mateo (2003)
16. Tan, K.C., Yu, Q., Ang, J.H.: A Dual-Objective Evolutionary Algorithm for Rules Extraction in Data Mining. Comput. Optim. Appl. 34, 273–294 (2006)
17. BLAKE, C.L. and C.J. MERZ, UCI repository of machine learning databases, (1998).

18. Moscato, Pablo. "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms." Caltech concurrent computation program, C3P Report 826 (1989): 1989

Classical Mechanics Optimization for image segmentation

Charaf eddine Khamoudj[1], Karima Benatchba[1], and Mohand tahar Kechadi[2]

[1] Laboratoire des Méthodes de Conception de Systèmes,
Ecole nationale Supérieure d'Informatique.
Oued smar, Algiers, Algeria
{c_khamouj, k_benatchba}@esi.dz
[2] School of Computer Science & Informatics,
Dublin, Ireland
{tahar.kechadi}@ucd.ie

Abstract. In this work, we focus on image segmentation by simulating the natural phenomenon of the bodies moving through space. For this, a subset of image pixels is regularly selected as planets and the rest as satellites. The attraction force is defined by Newton's third law (gravitational interaction) according to the distance and color similarity. In the first phase of the algorithm, we seek an equilibrium state of the earth-moon system in order to achieve the second phase, in which we search an equilibrium state of the earth-apple system. As a result of these two phases, bodies in space are constructed; they represent segments in the image. The objective of this simulation is to find and then extract the multiple segments from an image.

Keywords: Image segmentation · Combinatorial Optimization · Artificial Intelligence · Metaheuristic · Classical Mechanics Optimization.

1 Introduction

Segmentation is an important step in the image processing; it extracts segments from images. Each segment represents a set of pixels (each pixel is defined by its coordinates and color).

Image segmentation can be seen as a combinatory optimization problem, because the goal is to find combinations of assigning pixels to segments. To find optimal partitioning in K groups of an n pixels image, all the possible partitions must be browsed. The number of possible partitions is given by the Stirling numbers of the second kind [1]:

$$S(n, k) = \frac{1}{k!}\sum_{i=1}^{k}(-1)^{k-i}\binom{k}{i}i^{n} \qquad \textit{Where:}\ \ \binom{k}{i} = \frac{k!}{i!(k-i)!} \tag{1}$$

If the optimal number of partitions is unknown, Stirling numbers are calculated for $k=1$ to $k=n$. The number of possible partitions is given by Bell number [1]:

$$B(n) = \sum_{k=1}^{n} S(n,k) \quad (2)$$

The Bell number quickly becomes very big (example: *B(10)=115975*). The heuristic approaches for solving a combinatorial problem is to find a good solution in a bounded time among an exponential number of possibilities. So they are based on finding a good compromise between the calculation time and the quality of the best solution found so far.

The objective is to use the state of bodies' equilibrium in the space as a heuristic to tackle the image segmentation problem. We have proposed and implemented an image segmentation method based on a new metaheuristic inspired by the natural phenomenon of the bodies' movement in space. The proposed metaheuristic is based on the impact of the attractive forces between the bodies during their movements.

To simulate this problem as a natural phenomenon of the bodies' movement in space, we need to define the planets, the satellites, and what the attraction force. For this, *m* pixels of the image are uniformly selected. These pixels represent the planets, the remaining pixels represent the satellites and the attraction force is defined by the color similarity and the distance between the planet and the satellite.

The earth-moon system equilibrium is to find a situation, in which every single satellite is in rotation over the planet that applies on it the strongest attraction force. The earth-apple system equilibrium is to find a situation, in which all the bodies are far from colliding on each other, the resulting bodies represent the segments of an image.

2 Metaheuristics inspired from the interaction force

In the universe, attraction forces are divided into two types: Gravity is an attractive force between the bodies, which depends on their masses. The electromagnetic interaction is an attractive force that acts on the elements with electrical charges. Some researchers have proposed metaheuristics based on the forces of attraction between bodies. These forces are generated either from the physical mass or the electric charge. Here are some examples of this type of metaheuristic:

2.1 Gravitational search algorithm (GSA)

The gravitational search algorithm [2] uses Newton's third law to calculate the forces of attraction and Newton's second law to deduce the speed of a body. The diversification of the search in ensured by attraction force; To intensify the search, the gravitational constant is linearly decreased with time. GSA algorithm is combined with Particle Swarm Optimization (PSO) to solve the image segmentation problem [3]; The result algorithm is used in the second phase to search for the optimal threshold estimation used as a search procedure in the first phase.

2.2 Charge Search System (CSS)

The search system based on the electric charge [4] is inspired by the electrostatic; attributing electrical charges to the particles. The algorithm is used as a step of local search to improve the founded solutions in PSO algorithm [5] to solve the image segmentation problem.

2.3 Gravitational Interactions Optimization (GIO)

Optimization by gravitational interactions [6] called particle swarm optimization with gravitational interactions. Each body stores its current position and its best position. The interactions of bodies follow the Newton's third law and move each body to a new location so that the whole population tends to reach the optimum. This method uses the Newton's second law to calculate the speed of a body. To intensify the search, authors use a mass unit placed in space to exert forces on other bodies to move them. When the bodies are close to each other, the resulting forces are strong, and there are many displacements..

2.4 Fusion-Fission metaheuristic

The fusion-fission metaheuristic [7] is inspired from nuclear physics. It is applied on the graph partitioning problem, the clustering of documents and image segmentation. The atom is formed of electrons with a negative charge and nucleons which form the atomic core. There are two kinds of Nucleons: protons, positively charged and neutrons, neutrally charged. The cohesion of the atomic core is ensured by their strong interactions. During the fission of an atom, the core divides into two fragments, along with several ejected neutrons. An atom can split either spontaneously if its core is too heavy, or because of being hit by a neutron. To merge, atoms must have sufficiently high speeds. He considers a cloud of nucleons. It is subjected to high temperature and pressure, so that the nucleons have great chances of collision. It is the fusion of these nucleons together that forms the resulting atoms, which will help achieve an equilibrium state of the system. Fission is used to explode the biggest or non stable atoms.

3 Classical Mechanics Optimization (CMO)

As mentioned earlier, metaheuristics based on the gravitational interaction are hybridized with other metaheuristics, such as GSA algorithm with simulated annealing, and GIO algorithm that is hybridized with the particle swarm optimization. The proposed method is independent; it relies on applying the laws of classical mechanics.

The CMO simulates the natural phenomenon of the bodies' movement in a space by considering the pixels as bodies. m of these pixels are selected as planets and the n remaining are considered satellites, the attraction force is defined by Newton's third law (gravitational interaction).

After the simulation of the problem as a system of bodies in space, we execute the algorithm in two main phases: The first phase is to find an equilibrium of the rotating satellites around planets by applying the earth-moon system. The second phase is to group the segments formed in the first phase by applying the earth-apple system.

3.1 Transformation of the problem into a system of bodies in space

The planets represent a subset S of the set E (E is the global set that contain all pixels of image), and satellites represent the subset N representing the complement of S in E. Rules (3) and (4) are to calculate the number of planets and the number of satellites:

$$m = PixelNbr \times \frac{PlanetNbr}{PlanetNbr + SatelliteN\ br} \tag{3}$$

$$n = Pixel Nbr - m \tag{4}$$

The following figure shows the image to segment, the black pixels represent the planets, the remaining are satellites.

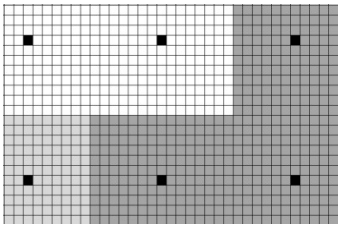


Fig. 1. Planets selection.

The number of pixels of each segment defines their mass:

$$\begin{aligned} &C\text{lassMass} = UnitMass \times ClassPixel\ Nbr \\ &\text{Where: } UnitMass = \frac{SystemMass}{PixelNbr} \end{aligned} \tag{5}$$

Newton's third law (gravitational interaction) is used to define the attraction force.

$$F_{ab} = g\frac{m_a m_b}{d_{ab}^{\ 2}} \tag{6}$$

Where m_a, m_b represent masses and d_{ab} represents distance between pixels a and b.

The distance d_{ab} is calculated by Euclidean distance, after the simulation of the spatial distance from a triangular rule:

GreatestPixelDistance(GPD) → *GreatestBodiesDistance(GBD)*

PixelDistance(PD) → *BodiesDistance(BD)*

So the distance ratio becomes:

$$\text{Distance Ratio} = \text{GBD/GPD} \tag{7}$$

The image is composed by a matrix of pixels; each pixel is defined by its coordinates and its color. After experimentation, the equation of attraction is improved as follows:

$$F_{ab} = \frac{m_a m_b}{d_{ab} e^{\sqrt{|c_a - c_b|}}} \tag{8}$$

Where c_a and c_b represent the color of the pixel a and the pixel b.
The gravitational fields earth-moon system GF_{em} and earth-apple system GF_{ea} are derived from the mechanic laws in rule (9) and rule (10) respectively:

$$GF_{em} = 1.068 \times mass \times 10^{-21} \tag{9}$$

$$GF_{ea} = GF_{\text{em}} / 200 \tag{10}$$

3.2 Finding a body equilibrium by applying the earth-moon system

We look in space for an equilibrium of the bodies, to stabilize the movement of satellites around planets. The movement of the satellites is caused by the gravitational attraction exerted by the planets.

A body a is rotating around the body b with a force F_{ab}. If there is a body c where: $F_{ac} > F_{ab}$, then the body a leave its path around b and follows a new path around c.

For each combination, we calculate the attraction force for planets. The gravity center becomes the center of all satellites around this planet.

We repeat the two previous steps until the system equilibrium is verified. The algorithm of this step is described as follows:

```
Algorithm 1 Find the system equilibrium {Earth-Moon sys-
tem}
  var    E: array [1..z,1..w] of real; {E is the image
             where z and w are the dimensions}
         Planet: array [1..m,1..n] of integer; {Each
            row of this matrix represents the satellites
            turned around the corresponding planet Since
            Planet(j, 1) represents the center of
            gravity of the same line}
         Satellite: array [1..n] of integer; {Each case i
            represents the corresponding planet of
            satellite i}
         m, n : integer; {m is planet number and n is
            satellite number }

  begin
    Calculate(m); Calculate(n);
    Initialize(Planet, Satellite);
```

```
    repeat
      For i = 1 to n
        For j = 1 to m
          If Earth-Moon gravitational field (Planet(j))
              > Distance (Planet(j),Satellite(i)) then
            If Force (Planet (j,1), Satellite(i)) > Force
               (Planet (Satellite (i),1),Satellite (i))
            then
              Move (Satellite(i), j);{Is to release the
                satellite i from his planet and assign it
                to the planet j}
            end;
          end;
        end;
      end;
    Until system stabilization
end.
```

The following figure shows a stable distribution of the satellites around the planets.

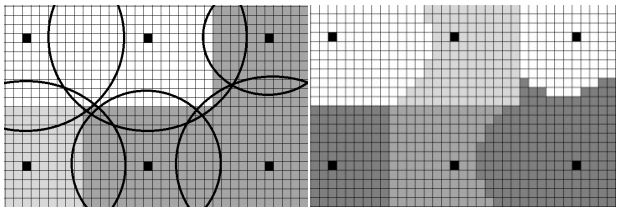


Fig. 2. System equilibrium for earth-moon system and grouping of bodies.

3.3 Construction of segments by applying the Earth-apple system

After the stabilization of satellites around planets, each planet-moon system is considered as one body. Then the gravitational fields of bodies (earth-apple system) is calculated. Each body situated in the gravitational field of another body falls (fusion of two segments) and the two bodies are considered as a single body. After the fusion, we repeat the previous two steps until the overall system is stable (all the found segments are too far to be fused). The algorithm of this phase is as follows:

```
Algorithm 2 Research earth-apple system equilibrium
  use Result of Algorithm 1
  begin
    repeat
      For i = 1 to n
        For j = 1 to m
          If Gravitational field earth-apple(Planet(i))
```

```
            > Distance(Planet(i), Planet(j)) then
          Move (Planet(j), i); {move all pixels of the
            body j to the body i, recalculate the new
            center of gravity and  remove the line j
            from the Planet Matrix}
          m := m-1;
        end;
      end;
    end;
  Until system stabilization
end.
```

The following figure shows the bodies gravitational fields and fusion (earth-apple system).

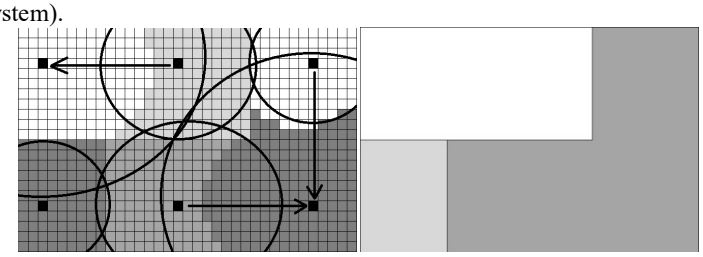


Fig. 3. Bodies gravitational fields and fusion (earth-apple system).

3.4 Tests and results

We applied the CMO approach on real images by simulating the solar system. The number of pixels planets *m* is calculated as follows:

*m = the number of pixels in the image*8 / (167 + 8)*

Because in the solar system there are one hundred and sixty seven (167) satellites and eight (08) planets. The results of the segmentation are:

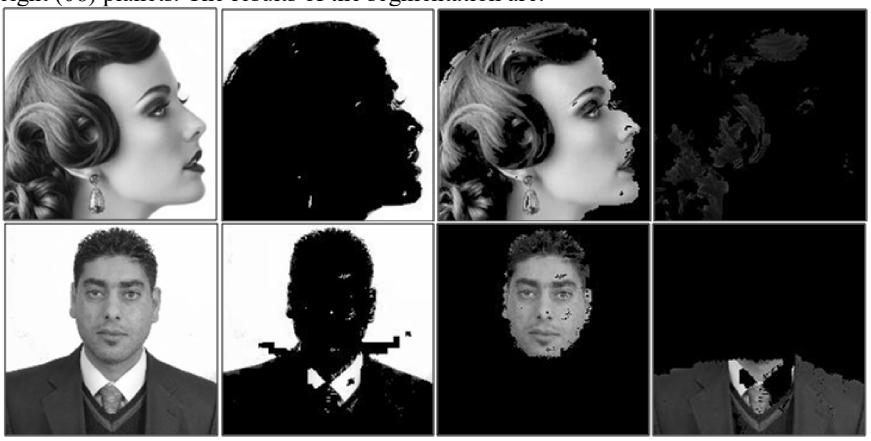


Fig. 4. Image segmentation results by using CMO.

The segmentation of the two images provides three segments that represent the objects of each image, that the dice coefficient of the approach is 88,65. There are small segments that are not displayed, it is the influence of light or stains. This is positive because these pixels or smaller segments can be treated isolated segments which are used to solve other problems such as the detection of tumors in medical imaging.

4 Conclusion

In this work, we developed an image segmentation method based on the simulation of the natural phenomenon of bodies' movement in space, called Classical Mechanics Optimization. It consisted on two phases. As a first step, we seek an equilibrium of the bodies in the earth-moon system (satellites assignment to the planets that apply more attraction force). The second step is to group the most similar segments, applying an earth-apple system.

The simulation is made by the extraction of a pixels subset as planets and the remaining pixels represent the satellites. The attraction force equation is defined by the rule (8) that represents Newton's third law according to the pixels' color, considering the importance of color in image segmentation.

In CMO, intensification and diversification are provided by the distance ratio that transforms the distance between pixels in the spatial distance. When it is small, the algorithm becomes more intensive because the gravitational field increases. However, if the distance ratio is very small, all the bodies may fall into a black hole which represents a segment that includes all pixels. Otherwise, if it is very large, grouping objects is not assured, because of the decrease in the gravitational field.

References

1. Benzaghou, B., Barsky, D.: Nombres de Bell et somme de factorielles. Journal de Théorie des Nombres de Bordeaux, N° 16, pages 1-17 (2004)
2. Rashedi, E., Nezamabadi, H., Saryazdi, S.: Gsa. A gravitational search algorithm. Information Sciences, 179(13): 2232-2248 (2009)
3. Amandeep, K., Charanjit, S., Amandeep, S.B.: SAR image segmentation based on hybrid PSOGSA optimisation algorithm ISSN: 2248-9622 Vol.4 Issue.9 (2014)
4. Barrera, J., Carlos, A.: A particle swarm optimization method for multimodal optimization based on electrostatic interaction. The 8th Mexican International Conference on Artificial Intelligence (MICAI '09), pages 622–632, Berlin, Heidelberg . Springer-Verlag (2009)
5. Dahiya, A., Dubey, R.B.: Survey of some multilevel thresolding techniques for medical imaging. ISSN: 2347-3878 Volume 3 Issue 7 (2015)
6. Flores, J.J., Farias, R.L., Barrera, J. : Particle swarm optimization with gravitational interactions for multimodal and unimodal problems. The 9th Mexican International Conference on Artificial Intelligence (MICAI 2010), pages 361–370. Springer- Verglas (2010)
7. Bichot, C.: Elaboration d'une nouvelle métaheuristique pour le partitionnement de graphe Doctoral thesis. The Polytechnic National Institute of Toulouse (2007)

Modern Heuristical Optimization Techniques for Power System State Estimation

Halil Alper Tokel, Gholamreza Alirezaei and Rudolf Mathar

Institute for Theoretical Information Technology, RWTH Aachen,
ICT cubes, Kopernikusstrasse 16, 52074 Aachen
{tokel,alirezaei,mathar}@ti.rwth-aachen.de
http://www.ti.rwth-aachen.de/

Abstract. The development of efficient and accurate algorithms for state estimation has come into the focus in power system research as the power grid becomes more decentralized. In this work, we apply the heuristical continuous optimization techniques *differential evolution*, *simulated annealing* and *particle swarm optimization* to power system state estimation problem, and provide a comparison between them in terms of convergence and optimality. Examining the results, we propose a hybrid algorithm combining particle swarm optimization and differential evolution.

Keywords: simulated annealing,particle swarm optimization,differential evolution,power system,meta-heuristic

1 Introduction

The accurate state estimation has been the most fundamental problem in power grids, since it delivers the system state as an input to all other applications in an energy management system. With the integration of renewable resources and the resulting decentralization of the power grids, an accurate and reliable information about the state of the system is required not only for the transmission level but also for the distribution level. This is a prerequisite to introduce new applications and to ensure a stable operation of the power system.

The traditional state estimation problem is formulated as a nonlinear weighted least square (WLS) problem, which can be solved iteratively by using gradient-based methods, e.g., the Gauss-Newton method. However, the success of these methods is based on a proper selection of a starting point, which is usually unknown. Thus, they often converge to a local minimum instead of a global one. On the other hand, the gradient is needed which is in practice replaced by an approximation, since the used objective functions and constraints are often discontinuous or very complicated to handle. In scenarios with a high number of distributed generators and consumers, this problem leads to Jacobian matrices [1], that are ill-conditioned. Considering the drawbacks of the basic state estimation approach, modern heuristical optimization techniques can provide an

alternative to deliver an accurate snapshot of the system state for the decentralized structure of future distribution grids.

Since the seminal work of Schweppe et.al. [2], numerous formulations of the state estimation problem and different numerical solution techniques have been proposed. In [3] a comprehensive survey of different state estimation techniques is given. The authors in [4] have proposed a hybrid particle swarm optimization(PSO) algorithm with a natural selection mechanism for evolutionary computation of the system state. For the computation, measurements consisting of branch voltages and injection values are used. Similarly, Mallick et.al. [5] have proposed a PSO algorithm with an additional differential evolution(DE) update step and have shown that the PSO performance can be improved by this method and it outperforms the Gauss-Newton method while considering ill-conditioned networks. Basetti et.al. [6, 7] have applied a gravitational search algorithm considering both traditional measurements and phasor measurement units (PMU), as well as a Taguchi differential evolution algorithm. Their proposed method delivers satisfactory results at the cost of longer computation time. In [8] a genetic algorithm-based technique is used, and the authors point out that the heuristical algorithm converges prematurely for IEEE 14 bus network without giving a good estimate. The authors of [9] use a self-adaptive evolutionary approach and conclude that evolutionary programming provides an accurate estimation in tests with IEEE test networks for 14 and 30 bus.

This literature review reveals the interest in power system research to find effective heuristical computational methods with good convergence properties in order to overcome the drawbacks of traditional numerical methods. It is well-known that the performance of heuristical techniques is highly dependent on the structure of the considered optimization problem [10]. Hence, for a specific optimization problem, one has to try out different heuristical approaches in order to find the one with the best performance. This is our main goal in the present work to compare all three methods, namely DE, simulated annealing(SA) and PSO, and provide meaningful results for the state estimation of energy grids.

We start with a description of the power system model as well as the traditional state estimation problem. Next, we introduce the optimization techniques followed by the description of the test cases along with simulation details. Finally, we present our main results and conclude our achievements.

2 Power System State Estimation

State estimation in power systems tries to obtain a reliable estimate of the voltage phasors at all system buses in the network by using a set of measurements. Although recent PMUs can measure voltage and current values with a high accuracy, the wide deployment of PMUs is costly, especially when considering the scale of distribution grids. For this reason, the traditional formulation of the state estimation problem still plays a crucial role.

In the following, we introduce the basic formulation of the state estimation problem for a power system. We define the system state $\mathbf{x}$ of a power grid with

n buses by the vector $\mathbf{x} = [\theta_2, \ldots, \theta_n, |V_1|, \ldots, |V_n|]$, where θ_i and $|V_i|$ are the angle and the magnitude of the voltage phasor at node i, respectively. The angle θ_1 of the first bus is set to zero and used as the reference angle. The set of measurements is denoted by

$$\mathbf{z} = \begin{bmatrix} h_1(\mathbf{x}) \\ \vdots \\ h_n(\mathbf{x}) \end{bmatrix} + \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix} = \mathbf{h}(\mathbf{x}) + \mathbf{e}, \tag{1}$$

where $h_i(\mathbf{x})$ is a nonlinear function, which describes the relation between the value of measurement i and the state vector $\mathbf{x}$, while $\mathbf{e}$ is the vector of measurement errors. It is assumed that the measurement errors are independent and zero-mean Gaussian distributed, i.e., $\mathbf{e} \sim \mathcal{N}(0, \mathbf{R})$, where $\mathbf{R} = \mathrm{diag}(\sigma_1^2, \ldots, \sigma_n^2)$ is the covariance matrix, with the variances σ_i^2 of the noise components e_i as its diagonal entries. The standard deviation σ_i of each measurement is modeled to take the accuracy of different measurements into account. The nonlinear measurement function $h_i(\mathbf{x})$ depends on the measurement type and location, and is formulated by the Kirchhoff rules for voltage and current. To illustrate the measurement function P_i for the real power injection at bus i, we may write

$$P_i = V_i \sum_{j=1}^{N} V_j (G_{ij} \cos\theta_{ij} + B_{ij} \sin\theta_{ij}), \tag{2}$$

where $\mathbf{G}$ and $\mathbf{B}$ are the real and imaginary parts of the bus admittance matrix $\mathbf{Y}$ of the power system and $\theta_{ij} = \theta_i - \theta_j$. The WLS estimator tries to find the state vector, which minimizes the error in the measurements. The optimization problem reads as

$$\underset{\mathbf{x}}{\text{minimize}} \;\; J(\mathbf{x}), \tag{3}$$

where $J(\mathbf{x})$ is defined by $(\mathbf{z} - \mathbf{h}(\mathbf{x}))^T \mathbf{R}^{-1} (\mathbf{z} - \mathbf{h}(\mathbf{x}))$. Note that the optimization problem (3) is subject to implicit constraints of state variables due to the assumption of a stable operation. Of course, a solution to (3) can be found by the Gauss-Newton method, however as mentioned in the introduction, the Gauss-Newton method has its own challenges. For a solid treatment of this approach please refer to [11].

3 Optimization Techniques

The heuristical techniques, which are considered in this work, are well-known methods which have been applied to different problems in engineering and other natural sciences. In this section, we introduce our notation for the power system state estimation problem and present the parameters adopted in this work. A good overview of modern heuristical techniques along with their applications in power systems can be found in [12].

3.1 Differential Evolution

In this work, we exploit the key ideas of differential variation, crossover and mutation in DE, where mutation and crossover operations occur with a certain probability, which is given by the algorithm parameter CR. The optimization variable $\mathbf{x}_i^k$ is encoded as a vector of floating point values.

The generation of a new candidate $\mathbf{x}_i^{k+1}$ follows the update equation

$$\mathbf{x}_i^{k+1} = \mathbf{x}_p^k + (\mathbf{x}_i^k - \mathbf{x}_{r_1})\lambda + (\mathbf{x}_{r_2} - \mathbf{x}_{r_3})F \tag{4}$$

where $\mathbf{x}_i^k$ is i^{th} member of the current generation k, $\mathbf{x}_p^k$ is the population member to perturb, λ and F are the crossover and mutation coefficients, respectively, and $\mathbf{x}_{r_i}$ are randomly selected population members. We select the first term $\mathbf{x}_p^k$ as the best population member $\mathbf{x}_{\text{best}}^k$. This strategy has outperformed other options in the test problems considered in this work.

3.2 Simulated Annealing

The key concepts in SA are cooling schedule, state generation and state acceptance. In present work, initial temperature is chosen as 100 and updated by $T_{k+1} = 0.95^k T_k$, where k is the iteration number. A new candidate $\mathbf{x}_{k+1}$ is generated by perturbing the current point $\mathbf{x}_k$ as in

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{r}\sqrt{T_k}, \tag{5}$$

where $\mathbf{r}$ is a random unit vector with $|\mathbf{r}| = 1$. The acceptance of a candidate with an inferior objective function value occurs with a probability calculated by the acceptance function f_{accept} as

$$f_{\text{accept}} = \exp\Big(\frac{J(\mathbf{x}_i) - J(\mathbf{x}_j)}{T_k}\Big). \tag{6}$$

Table 1. Summary of algorithm parameters and functions in this work

Algorithm	Parameters	Functions
DE	$CR = 0.7$ $F = 0.7$, $\lambda = 0.5$	$\mathbf{x}_i^{k+1} = \mathbf{x}_{\text{best}}^k + (\mathbf{x}_i^k - \mathbf{x}_{r_1})\lambda + (\mathbf{x}_{r_1} - \mathbf{x}_{r_2})F$
SA	$T_0 = 100$	$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{r}\sqrt{T_k}, \|\mathbf{r}\| = 1$ $T_{k+1} = 0.95^k T_k$, $f_{\text{accept}} = \exp\Big(\frac{J(\mathbf{x}_i) - J(\mathbf{x}_j)}{T_k}\Big)$
PSO	$c_{1,2} = 1.49$ $w = [0.4, 0.8]$	$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + (\mathbf{p}_{\text{best},i} - \mathbf{x}_i^k)c_1 r_1 + (\mathbf{g}_{\text{best},i} - \mathbf{x}_i^k)c_2 r_2$ $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k$

3.3 Particle Swarm Optimization

In PSO, a population with N members have their positions $\mathbf{x}_i^k$ and velocities $\mathbf{v}_i^k$ where k is the iteration number and i is the member index. Each member knows its personal best value $\mathbf{p}_{\text{best},i}$ and the best value of its neighborhood $\mathbf{g}_{\text{best},i}$. The velocity $\mathbf{v}_i$ and the position $\mathbf{x}_i$ are updated by

$$\mathbf{v}_i^{k+1} = w\mathbf{v}_i^k + (\mathbf{p}_{\text{best},i} - \mathbf{x}_i^k)c_1 r_1 + (\mathbf{g}_{\text{best},i} - \mathbf{x}_i^k)c_2 r_2, \tag{7}$$

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \mathbf{v}_i^k, \tag{8}$$

where w is called the weight, c_1, c_2 are cognitive and social acceleration coefficients, and r_1 and r_2 are independent and uniformly distributed random numbers between 0 and 1. If the new member $\mathbf{x}_i^{k+1}$ achieves a smaller value of the objective function, the member and its personal best are updated.

In Table 1 the chosen parameters for all algorithms are summarized.

4 Test Cases

In this section, we describe the test networks considered in the present work. We use the IEEE 14 and 30-bus test networks [13] for the comparison of the heuristical optimization techniques whose details are provided in the previous section. In this work, we consider only power flow and generation injection measurements. The power flow measurements are located on the branches of a spanning tree of the networks. With the injection measurements, this placement ensures full observability [11]. The exact measurement locations are listed in Table 2.

Table 2. Measurements used for calculations in IEEE 14- and 30-bus test networks

	Power Flow on Branches	Generation at Buses
14-Bus	1-2,1-5,2-3,2-4,4-7,4-9,5-6,6-11,6-12,6-13,7-8,9-10,9-14	1,2,3,6,8
30-Bus	1-2,1-3,2-4,3-4,2-5,2-6,6-7,6-8,6-9,6-10,23-24,25-26, 6-28,9-11,10-17,10-20,10-21,10-22,12-13,25-27,28-27, 12-15,12-16,14-15,15-18,15-23,18-19,22-24,27-29,27-30	1,2,5,8,11,13

We use the MATLAB package MATPOWER 5.1 to obtain the real (correct) measurement values and the values of the state variables by solving the optimal power flow problem [14]. The measurement values are then overlaid with additive Gaussian noise with the standard deviation σ_i of 0.02 and 0.015 for power flow and injection measurements, respectively.

Two different approaches are used for the initialization of the first candidates. In the first approach, we set the starting candidate in SA and one member of the first populations of DE and PSO to a candidate obtained by overlaying the true system state along with a noise which has the same statistics as in the measurement creation step. In other words, the algorithms start from a point in

the search space which is close to the true system state. This approach, which we call *near start* in the following, is reasonable since the power system state changes gradually and the result of the last estimation can be used as the starting point in the next estimation. For the sake of completeness, in the second approach, we apply the same procedure with a flat start, where all voltage phasor magnitudes $|V_i|$ are set to 1 p.u. and all voltage phasor angles θ_i to 0°. In both approaches, other candidates in DE and PSO are randomly generated over the search space. For each optimization algorithm, we perform Monte Carlo simulations with 30 runs with an iteration limit of 1000 iterations.

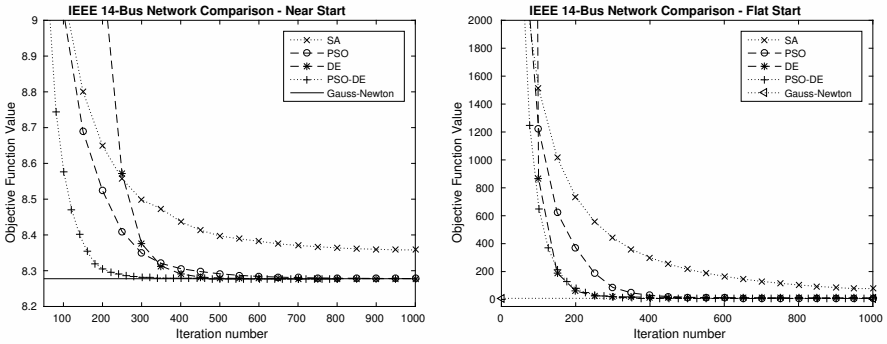


Fig. 1. Values of the objective function at certain iteration points for the IEEE 14-bus network for all three heuristical optimization techniques DE, SA, PSO and hybrid PSO-DE algorithm. Left: Near Start, Right: Flat Start.

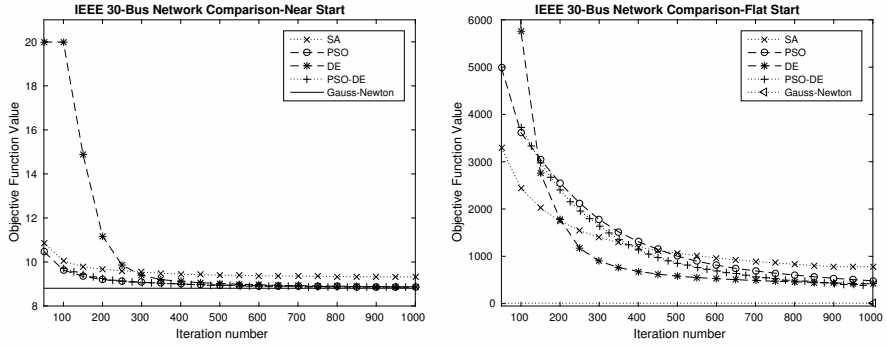


Fig. 2. Values of the objective function at certain iteration points for the IEEE 30-bus network for all three heuristical optimization techniques DE, SA, PSO and hybrid PSO-DE algorithm. Left: Near Start, Right: Flat Start.

5 Results

The results for IEEE 14-bus and 30-bus test networks are illustrated in Figure 1 and Figure 2, respectively. The graphs on the left provide a comparison of the expected values of the objective function for near start initializations, whereas the graphs on the right side show the results with flat start. It is noticeable in near start case in Figure 1 that PSO performs best in first iterations, but is overtaken by DE after 300 iterations. This observation has led us to propose an hybrid solution with PSO and DE to benefit from their superior performances in search and intensification, respectively. In this algorithm, we observe the rate of decrease in the best value of PSO until it reaches a threshold value. After the threshold is reached, the algorithm continues with the update steps of DE. We set the threshold to 5% of the improvement in the first 5 iterations, where the decrease in the best function value is compared with the threshold at every five iterations. As can be seen in Figure 1, the proposed hybrid solution outperforms all other algorithms considerably in near start case. On the other hand, we observe a very slight improvement in flat start initialization. In fact, DE outperforms PSO in flat start initialization, which can be attributed to the start from a worse population and the decrease in the search capability of PSO with increasing iteration number. Another observation is the inferior performance of SA compared with PSO and DE.

In Figure 2, the results of 30-bus test case enable the evaluation of the scalability of the algorithms in a larger problem size. We observe in near start initialization that the hybrid algorithm improves the performance of PSO marginally. This is reasonable as DE does not outperform PSO in later iterations as in 14-bus test case. In flat start case, we see that none of the algorithms can achieve acceptable objective function values comparable to the result of Gauss-Newton method, which can be reasoned by the larger problem size and the iteration limit of the algorithms. Interesting is that the hybrid algorithm improves the PSO performance only slightly, although the performance of DE is better than PSO in flat start initialization. Regarding the variation in the achieved results, PSO, DE and the hybrid algorithm have an average normalized standard deviation of 0.08, 0.1, and 0.09% in 14-bus near start case, and 0.26, 0.41, and 0.23% in 30-bus near start case, respectively. On the other hand, the variation in the flat start case is as high as 36 and 89% in 14- and 30-bus networks, respectively.

6 Conclusion

In this work, we have applied the modern heuristical optimization techniques DE, SA, and PSO to the problem of state estimation in power systems with traditional measurements. Based on the comparison results in 14-bus network test case, we have proposed a hybrid algorithm with PSO and DE, which has improved the convergence noticeably. On the other hand, we have deduced that a larger problem size poses a challenge for the hybrid algorithm, whereas none of the algorithms can deliver satisfactory results with flat start initialization.

Nevertheless, the decentralized structure of future distribution grids can enable the use of heuristical techniques in a distributed manner.

References

1. Gu, J. W., Clements, K. A., Krumpholz, G. R., Davis, P. W.: The Solution of Ill-Conditioned Power System State Estimation Problems Via the Method of Peters and Wilkinson. IEEE Transactions on Power Apparatus and Systems, vol. PAS-102, no. 10, pp. 3473-3480 (1983)
2. Schweppe, F. C., Wildes, J.: Power System Static-State Estimation, Part I: Exact Model. IEEE Transactions on Power Apparatus and Systems, vol. PAS-89, no. 1, pp. 120-125 (Jan. 1970)
3. Hayes, B., Prodanovic, M.: State Estimation Techniques for Electric Power Distribution Systems. In Proc. of 2014 European Modelling Symposium(EMS), pp. 303-308 (2014)
4. Naka, S., Fukuyama, Y., Genji, T., Yur, T.: A Practical Distribution State Estimation Using Hybrid Particle Swarm Optimization. In Proc. of IEEE Power Engineering Society Winter Meeting, vol. 1, pp. 571-575, Ohio, USA (28 Jan-1 Feb 2001)
5. Mallick, S., Ghoshal, S. P., Acharjee, P., Thakur, S. S.: Optimal Static State Estimation Using hybrid Particle Swarm-Differential Evolution Based Optimization. Journal of Energy and Power Engineering, Vol. 5, No. 4B, pp. 670-676 (2013)
6. Basetti, V., Chandel, A. K.: Power system state estimation using gravitational search algorithm. In International Conference on Computer and Computational Sciences (ICCCS), pp. 32-38. Noida (27-29 Jan. 2015)
7. Basetti, V., Chandel, A. K.: Hybrid power system state estimation using Taguchi differential evolution algorithm. Journal of Science, Measurement & Technology, IET, vol. 9, no. 4, pp. 449-466 (2015)
8. Hossam-Eldin, A. A., Abdallah, E. N., El-Nozahy, M. S.: A Modified Genetic Based Technique for Solving the Power System State Estimation Problem. Journal of World Academy of Science, Engineering & Technology, Issue 31, p. 311 (2009)
9. Contreras-Hernandez, E. J., Cedeno-Maldonado, J. R.: A Self-Adaptive Evolutionary Programming Approach for Power System State Estimation. In 49th IEEE International Midwest Symposium on Circuits and Systems MWSCAS'06, vol. 1, pp. 571-575 (6-9 Aug. 2006)
10. Navarro, R., Puris, A., Bello, R.: The Performance of Some Meta-heuristics in Continuous Problems Studied According to the Location of the Optima in the Search Space. Dyna rev.fac.nac.minas [online]. 2013, vol. 80, n. 180 [cited 2016-02-09], pp. 60-66 . Available from: `http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532013000400008&lng=en&nrm=iso`.
11. Abur, A., Expósito, A. G.: Power System State Estimation: Theory and Implementation. CRC Press, New York (2004)
12. Lee, K.Y., El-Sharkawi, M.A.: Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems. Wiley-IEEE Press, Hoboken, New Jersey(2008)
13. Power Systems Test Case Archive, `https://www.ee.washington.edu/research/pstca/`
14. Zimmerman, R. D., Murillo-Sánchez, C. E., Thomas, R. J.: MATPOWER: Steady-State Operations, Planning and Analysis Tools for Power Systems Research and Education. IEEE Transactions on Power Systems, vol. 26, no. 1, pp. 12-19 (Feb. 2011)

On the community identification in weighted time-varying networks

Youcef Abdelsadek[1,2] *, Kamel Chelghoum[1], Francine Herrmann[1], Imed Kacem[1], and Benoît Otjacques[2]

[1] Laboratoire de Conception, Optimisation et Modélisation des Systèmes
Université de Lorraine, Metz, France,
{youcef.abdelsadek,kamel.chelghoum,
francine.herrmann,imed.kacem}@univ-lorraine.fr
[2] e-Science Research Unit, Environmental Research and Innovation Luxembourg
Institute of Science and Technology, Belvaux, Luxembourg
{youcef.abdelsadek,benoit.otjacques}@list.lu

Abstract. The community detection play an important role in understanding the information underlying to the graph structure. Especially, when the graph structure or the weights between the linked entities change over time. In this paper, we propose an algorithm for the community identification in weighted and dynamic graphs, called *Dyci*. The latter takes advantage from the previously detected communities. Several changes' scenarios are considered like, node/edge addition, node/edge removing and edge weight update. The main idea of *Dyci* is to track whether a connected component of the weighted graph becomes weak over time, in order to merge it with the "dominant" neighbour community. In order to assess the quality of the returned community structure, we conduct a comparison with a genetic algorithm on real-world data of the ARN-Info-RSN project. The conducted comparison shows that *Dyci* provides a good trade-off between efficiency and consumed time.

Keywords: dynamic networks, community detection, genetic algorithm, weighted graphs, Twitter's networks.

1 Introduction

With the popularization of social networks like Twitter, an exponential quantity of data is generated. These data are increasing each day, and the existing algorithms which are not considering the dynamism nature of data would suffer from the scalability issue. Furthermore, the community detection in a dynamic network enhances our understanding of the underlying dynamic graph. The changes that might occur can be, either the structure of the graph, its attributes or also both of them. Consequently, how to analyse the evolution of the communities structure over time? To answer this question, one need to devise an algorithm which takes advantage from the previously identified communities by avoiding

* Corresponding author

the community identification from scratch at each instant. As a concrete example, an analyst needs to understand how the information is shared in Twitter. To fulfil this need, one have to detect the communities of the analyst's time point of interest and to follow the community's member evolution with new members joining/leaving the studied communities. In this context, a trade-off between efficiency and response time is necessary.

A dynamic graph of an initial graph G_0 can be seen as a sequence of static graphs [5], denoted by $G_s = (G_0, G_1, \ldots, G_f)$ with f snapshots giving rise to $Cs_s = (Cs_0, Cs_1, \ldots, Cs_f)$ community partitions as results of $U_s = (U_0, U_1, \ldots, U_{f-1})$ updates as illustrated in Figure 1. In [6] the authors introduces the properties of a dynamic graph. Those can be divided into two categories: the structural category and the attributes category. We denote by N_t, E_t, E_t^w, N_s and E_s respectively, the set of nodes at instant t, the set of edges at instant t, the set of edge weights at instant t of G_t, the set of nodes of the whole G_s and the set of edges of the whole G_s. Furthermore, the set of updates U_t varies in terms of the impact they cause to the current set of communities.

1. **Structural updates**:
   (a) **Node removing**: An old node on is removed, $N_{t+1} \leftarrow N_t \setminus \{on\}$ with the related edges.
   (b) **Edge removing**: An old edge oe is removed, $E_{t+1} \leftarrow E_t \setminus \{oe\}$.
   (c) **Node addition**: A new node nv is added, $N_{t+1} \leftarrow \{nv\} \cup N_t$ with the related edges.
   (d) **Edge addition**: A new edge ne is added, $E_{t+1} \leftarrow \{ne\} \cup E_t$.
2. **Attributes updates**:
   (a) **Edge weight updating**: An new edge weight ne^w of an old edge weight oe^w is updated, $E_{t+1}^w \leftarrow (E_t^w \setminus \{oe^w\}) \cup \{ne^w\}$.

The outlines of the remaining sections of this paper are as follow: In Section 2 the related work of the addressed topic are presented. Section 3 introduces the *Dyci* algorithm for community identification in weighted and dynamic graphs. Section 4 describes the genetic algorithm. In section 5 the conducted assessments and the obtained results are discussed. Finally, Section 6 concludes this paper.

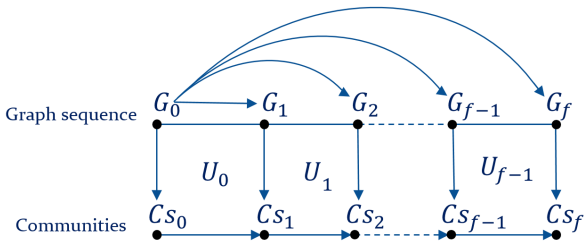


Fig. 1: The graph sequence of a dynamic graph

2 Related work

This section presents some relevant works for the dynamic community detection. There are more algorithms for the static version of this problem in the literature compared to the dynamic case, especially, for those considering weighted edges. In [13], the dynamic community detection problem was proved to be NP-complete and APX-hard. For the unweighed dynamic community detection, in [12] the authors propose a matching algorithm to detect similar communities over the snapshots, introducing a meta-community which is the sequence of these identical communities. Agglomerative modularity-based approach are considered in [4], [2], [9]. The authors of [9] uses a physical metaphor with forces which retains a node to stay in its community against attracting forces of the other communities. Furthermore, game-theoretic analogy is used in [3]. In the latter, each node of the graph is considered as an agent which maximizes its utility function. A set of predefined agent actions is set initially. The system ends when all agents choose their best community belonging (i.e., which maximizes the utility). Finally, relying on the colouring problem, a constant-approximation algorithm was proposed in [14]. Regarding the weighted version of this problem, label propagation is used [15]. The idea of this algorithm is to allow a specific node to change its community label taking into account its adjacent nodes labels.

3 Dynamic community detection algorithm

3.1 Notations and definitions

Let us define c_{n_i}, IW, INW, WD and WI which, respectively, represent the community of the node n_i, the intra-community weight, the inter-community weight, the weighted degree of a node and the weighted community-incidence of a node. These are presented in the following equations:

$$IW_{c_g} = \sum_{i<j} e^w_{n_i,n_j}, \forall n_i, \forall n_j \in c_g \tag{1}$$

$$INW_{c_g,c_h} = \sum_{i<j} e^w_{n_i,n_j}, \forall n_i \in c_g, \forall n_j \in c_h \tag{2}$$

$$WD_{n_i} = \sum_{j=1}^{v} e^w_{n_i,n_j} \tag{3}$$

$$WI(node, c_g) = \frac{\sum e^w_{node,n_i}}{WD_{node}} \text{ such as } n_i \in c_g \tag{4}$$

3.2 *Dyci* algorithm

An improved version of the algorithm proposed in [1] is applied on G_0 as a starting point Cs_0 of *Dyci*.

1. **Node removing (*oldNode*):** The main idea of the node removing case is to check whether the deletion of $oldNode$ either generates several connected components or reduces the $IW(c_{oldNode})$. To this end, *Dyci* tests for each resulting connected component, noted CC, whether it can form a community by it self or would be merged with an adjacent community, noted *com*. In other words, *Dyci* verifies whether Equation 5 holds or not.

$$INW_{com,CC} \geqslant IW_{CC} \tag{5}$$

2. **Edge removing (*oldEdge*):** When an inter-community edge is removed, this reduces the inter-community weight leading to more community-like structure. However, the other case might lead to intra-community dividing in two connected components or a significant weight loss. To handle this case, *Dyci* compares weights between each resulting connected component of *oldEdge* deletion and their adjacent communities by Equation 5.
3. **Node addition (*newNode*):** Two subcases can occur for node addition. The first one is the subcase where $newNode$ has no community edge incidence leading to an isolated community. In the second subcase, $newNode$ comes with many edges. For the latter, $newNode$ is added to the community with the greatest $WI(newNode, c), \forall c \in$ communities adjacent to $newNode$.
4. **Edge addition (*newEdge*):** For this case, if *newEdge* is inserted inside a community, this will not affect the community partition in terms of weights. Unlike, an inter-community edge could increase the inter-community weight, noted c_1 and c_2, aggregating them in one community. To handle this case, *Dyci* verifies whether Equation 6 holds or not.

$$INW_{c_1,c_2} \geqslant IW_{c_1} \text{ or } INW_{c_1,c_2} \geqslant IW_{c_2} \tag{6}$$

5. **Edge weight updating (*edgeWeightUpdate*):** The last case be partitioned into two subcases. The first is when the *edgeWeightUpdate* is an inter-community edge with weight greater than the old edge weight. For this scenario *Dyci* verifies whether Equation 6 holds or not. The second subcase rises when *edgeWeightUpdate* is an intra-community edge with weight lower than the old edge weight. The algorithm checks whether this weight loss leads to an adjacent community merging by Equation 5.

4 Genetic algorithm

Genetic algorithms (GA) can provide very good results if they are well set. In order to evaluate the quality of the obtained communities of Cs_f, a comparison is conducted between *Dyci* and the following GA.

– **Chromosome encoding**: The Locus-based Adjacency Representation [10] (LAR) is used to encode the community detection problem, like in [11], [7]. In the LAR a $|N_f|$ sized array is used, where the couples (gene, allele) express an associative community membership. Indeed, each gene takes its allele value from the set of its node neighbours ensuring feasible solutions. Figure 2 shows an example with the related individual decoding.

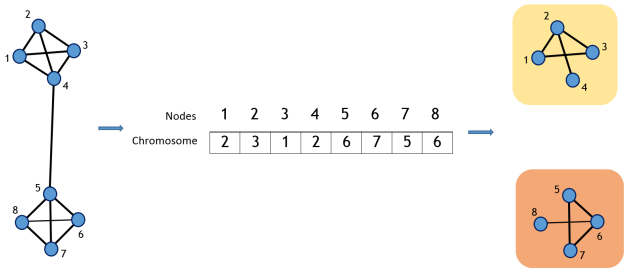
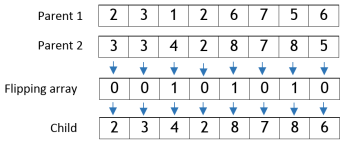
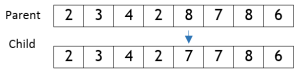


Fig. 2: An individual example using LAR encoding



(a) Uniform crossover



(b) Mutation

Fig. 3: Reproduction operators

– **Fitness function**: Modularity φ of [8] is used for individual evaluation:

$$\varphi = \frac{1}{2M} \sum_{n_i} \sum_{n_j} \left(e^{w}_{n_i,n_j} - \frac{WD_{n_i} WD_{n_j}}{2M} \right) \delta\left(c_{n_i}, c_{n_j}\right) \tag{7}$$

$$\text{Such as, } M = \sum_{i<j} e^{w}_{n_i,n_j} \text{ and } \delta\left(c_{n_i}, c_{n_j}\right) = 1, \text{ if } c_{n_i} = c_{n_j}, 0 \text{ otherwise.}$$

– **Population initialization**: A random population of size 100 is generated and sorted in a decreasing fitness function order.
– **Crossover**: Uniform crossover with probability 0.9 is performed, as illustrated in Figure 3a.
– **Mutation**: Random allele flipping with probability 0.1 is performed, as showed in Figure 3b.
– **Parent selection and child insertion**: Random selection from the 20% eliteness individuals. Weakest individuals are excluded from the population.
– **Stopping condition**: Number of generations reaches 50.

5 Experiments and results discussion

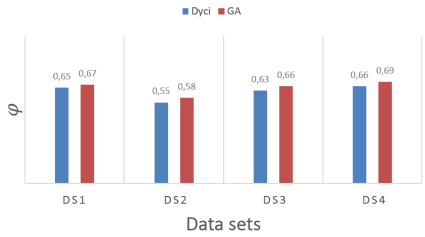
This section discusses the obtained results of the conducted comparison between the above GA and *Dyci*. Four datasets from real-world data of the ANR-Info-RSN project are considered. The ANR-Info-RSN project deals with the community detection in Twitter's network. To this end, the dynamic graphs model

the re-tweets between Twitter's users over time. Table 1 presents the dataset characteristics. The unit of snapshot generation is one day. Figure 4 and Figure 5 show, respectively, the obtained results for the datasets at t_f and the averages values of the results for the datasets for the whole Cs_s.

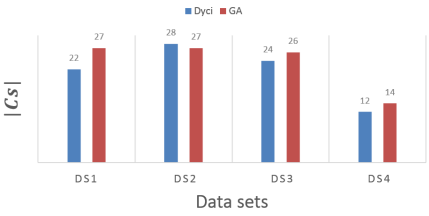
From Figure 4a, we remark that *Dyci* and the GA have almost the same results (GA very slightly better). By taking into account the fact that the obtained communities Cs_f of *Dyci* are highly influenced by the f previous choices made, one could say that *Dyci* obtains satisfactory results. Further, from Figure 4c, we remark that *Dyci* is relatively fast compared to the GA. Indeed, *Dyci* takes advantage from the previous identified community avoiding relaunching process at each snapshot. From Figure 5, we notice that the averages values are almost the same comparing to the values of the last snapshot t_f, except for DS3 where *Dyci* takes more time and provides less modularity for the previous snapshots but has relatively good result for the last snapshots t_f.

Table 1: The ANR-Info-RSN datasets characteristics

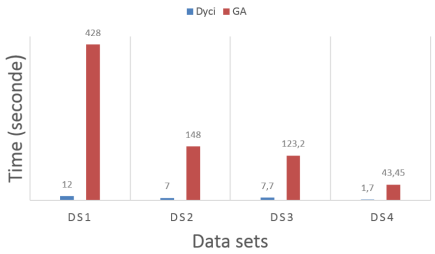
Data sets	t_0	t_f	$\|N_s\|$	$\|E_s\|$	$\|N_f\|$	$\|E_f\|$
DS1	July 17,2014	July 31,2014	10569	14121	801	997
DS2	August 3,2014	August 15,2014	6162	8069	390	451
DS3	August 17,2014	August 31,2014	10189	12263	424	508
DS4	September 3,2014	September 30,2014	8224	10371	412	535



(a) Obtained modularity

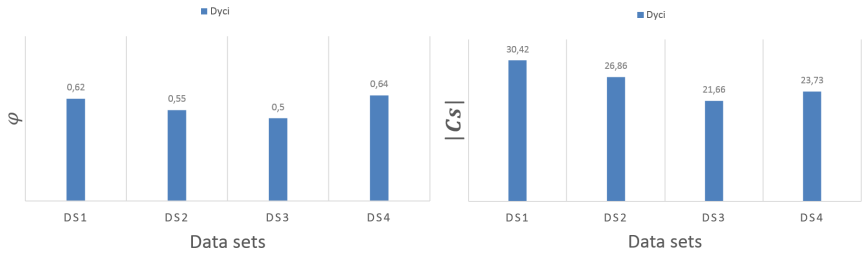


(b) Identified number of communities



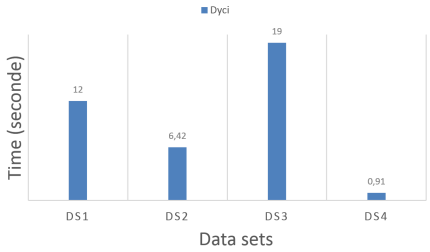
(c) Running time

Fig. 4: The results for the ANR-Info-RSN datasets at t_f



(a) Obtained modularity

(b) Identified number of communities



(c) Running time

Fig. 5: The results for the ANR-Info-RSN datasets for the whole Cs_s

6 Conclusion

To conclude, community identification in time-varying networks enhances our understanding of the graph structure over time. In this paper, a community detection algorithm for weighted and dynamic graphs, called *Dyci*, is proposed. The main idea of *Dyci* is to track whether a connected component of the weighted graph becomes weak (i.e., in terms of weight) over time, in order to merge it with the "dominant" neighbour community. In order to assess the quality of the identified communities by *Dyci*, a computational comparison is conducted with GA on real-world datasets of the ANR-Info-RSN project. The latter shows that *Dyci* obtains satisfying results with relatively short time.

Acknowledgments. This research has been supported by the Agence Nationale de la Recherche (ANR, France) during the Info-RSN Project (ANR-13-SOIN-0008).

References

1. Youcef Abdelsadek, Kamel Chelghoum, Francine Herrmann, Imed Kacem, and Benoît Otjacques. Community detection algorithm based on weighted maximum triangle packing. In *Proceedings of International Conference on Computer and Industrial Engineering CIE45*, 2015.

2. Riza Aktunc, Ismail Hakki Toroslu, Mert Ozer, and Hasan Davulcu. A dynamic modularity based community detection algorithm for large-scale networks: Dslm. In Jian Pei, Fabrizio Silvestri, and Jie Tang, editors, *ASONAM*, pages 1177–1183. ACM, 2015.
3. Hamidreza Alvari, Alireza Hajibagheri, and Gita Reese Sukthankar. Community detection in dynamic social networks: A game-theoretic approach. In Xindong Wu, Martin Ester, and Guandong Xu, editors, *ASONAM*, pages 101–107. IEEE Computer Society, 2014.
4. Shweta Bansal, Sanjukta Bhowmick, and Prashant Paymal. Fast community detection for dynamic complex networks. In Luciano da F. Costa, Alexandre Evsukoff, Giuseppe Mangioni, and Ronaldo Menezes, editors, *CompleNet*, volume 116 of *Communications in Computer and Information Science*, pages 196–207. Springer, 2010.
5. Stephan Diehl and Carsten Görg. *Graph Drawing: 10th International Symposium, GD 2002 Irvine, CA, USA, August 26–28, 2002 Revised Papers*, chapter Graphs, They Are Changing, pages 23–31. Springer Berlin Heidelberg, Berlin, Heidelberg, 2002.
6. F. Harary and G. Gupta. Dynamic graph models. *Math. Comput. Model.*, 25(7):79–87, April 1997.
7. Di Jin, Dongxiao He, Dayou Liu, and Carlos Baquero. Genetic algorithm with local search for community mining in complex networks. In *ICTAI (1)*, pages 105–112. IEEE Computer Society, 2010.
8. MEJ Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.
9. Nam P. Nguyen, Thang N. Dinh, Ying Xuan, and My T. Thai. Adaptive algorithms for detecting community structure in dynamic social networks. In *INFOCOM*, pages 2282–2290. IEEE, 2011.
10. YoungJa Park and ManSuk Song. A genetic algorithm for clustering problems. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 568–575, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.
11. Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. In *PPSN*, volume 5199 of *Lecture Notes in Computer Science*, pages 1081–1090. Springer, 2008.
12. Mansoureh Takaffoli, Farzad Sangi, Justin Fagnan, and Osmar R. Zane. Community evolution mining in dynamic social networks. *Procedia - Social and Behavioral Sciences*, 22:49 – 58, 2011. Dynamics of Social Networks7th Conference on Applications of Social Network Analysis - {ASNA} 2010.
13. Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. A framework for community identification in dynamic social networks. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 717–726, New York, NY, USA, 2007. ACM.
14. Chayant Tantipathananandh and Tanya Y. Berger-Wolf. Constant-factor approximation algorithms for identifying dynamic communities. In John F. Elder IV, Franoise Fogelman-Souli, Peter A. Flach, and Mohammed Zaki, editors, *KDD*, pages 827–836. ACM, 2009.
15. Jierui Xie, Mingming Chen, and Boleslaw K. Szymanski. Labelrankt: Incremental community detection in dynamic networks via label propagation. *CoRR*, abs/1305.2006, 2013.