A Self-adaptive Multiagent Evolutionary Algorithm for Electrical Machine Design

Jean-Laurent Hippolyte
Laboratoire d'Informatique de l'Université de Franche-Comté
Centre Numerica
1, cours Leprince-Ringuet
25200 Montbéliard
France
jean-laurent.hippolyte@lifc.univ-fcomte.fr

Christelle Bloch
Laboratoire d'Informatique de l'Université de Franche-Comté
Centre Numerica
1, cours Leprince-Ringuet
25200 Montbéliard
France
bloch@lifc.univ-fcomte.fr

Pascal Chatonnay
Laboratoire d'Informatique de l'Université de Franche-Comté
Centre Numerica
1, cours Leprince-Ringuet
25200 Montbéliard
France
pascal.chatonnay@univ-fcomte.fr

Christophe Espanet
Laboratoire d'Electronique, Electrotechnique et Systèmes
UTBM-L2ES (Bat F)
Rue Ernest Thierry-Mieg
90010 Belfort Cedex
France
christophe.espanet@univ-fcomte.fr

Didier Chamagne
Laboratoire d'Electronique, Electrotechnique et Systèmes
UTBM-L2ES (Bat F)
Rue Ernest Thierry-Mieg
90010 Belfort Cedex
France
didier.chamagne@univ-fcomte.fr

ABSTRACT

This paper presents a self-adaptive algorithm that hybridises evolutionary and multiagent concepts. Each evolutionary individual is implemented as a simple agent capable of reproduction and predation. The transitions between these two states depend on the agent's local environment. Thus, no explicit global process is defined to select neither the mates nor the preys. The convergence of the algorithm emerges from the behaviour of the agents. This brings interesting properties, such as population size self-regulation. Two sets of experimental results are provided: a comparison with Saw-Tooth Algorithm [5] and μGA [3] using four classical functions and an optimisation of the efficiency and the weight of an electrical motor. Some possible evolutions and prospects are finally proposed.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems*; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms

Keywords

optimisation, genetic algorithm, multiagent system, self adaptation

1. INTRODUCTION

Unlike classical GAs, the presented algorithm lets the genetic individuals be the actors of the optimisation. An individual becomes an agent that determines which action it has to accomplish, reproduction or predation, without the guidance of an other entity of the program. Classical GAs are also often criticised for difficult parameter adjustment, that's why some recent researches focus on developing self-adaptive algorithms [7]. The presented algorithm takes advantage of both multiagent systems (MAS) and self-adaptive genetic algorithms.

1.1 Multiagent Evolutionary Algorithms

There are two types of algorithms or frameworks that associate MAS and GA. On the one hand, there are frameworks where an agent is an instance of meta-heuristics [11] [6] and on the other hand there are algorithms where an agent is an instance of solution. In 2004, Milano and Roli [6] presented MAGMA (multiagent metaheuristic Architecture) . These authors define an agent as a system able to build a solution, to move over a landscape, to communicate with other agents, to be active (i.e. goal oriented) and possibly, to be adaptive. This way, new algorithms (that are combinations of metaheuristics) can easily be designed by choosing the agents to be involved and by defining their interactions.

In 2006, Talbi and Bachelet [11] proposed a parallel cooperative metaheuristic, named Cosearch. This framework is based on three heuristic agents implemented on a parallel environment to result in a well-balanced metaheuristics, by using their complementary behaviours. The main search algorithm is a Multiple Tabu search, the diversifying agent is a GA and a kick operator is used as an intensifying agent.

In 2004, Zhong et al. [12] integrated MAS and GAs to form a new algorithm, named multiagent Genetic Algorithm (MAGA). This approach is based on agents that correspond to solutions and live in a lattice-like environment. Each agents aims at increasing its own energy by cooperating or competing with its four neighbours. Four evolutionary operations may be performed on an agent: competition, cooperation, mutation and self-learning. Competition uses a kind of tournament to identify whether the agent is a loser or not. If it is a loser it is replaced by a new agent, generated either by a heuristic crossover or by an inversion operation. Cooperation is based on a neighbourhood orthogonal crossover. Self-learning is a local search based on a small scale MAGA performed on the best agent at each generation.

MAGMA and COSEARCH are different from GMAS (Genetic multiagent System), the approach proposed in this paper. In the former approaches, agents are instances of metaheuristics, whereas GMAS agents are instances of solutions. MAGA is more similar to GMAS: in both algorithms, an agent represents a solution, and there is no global selection. But in MAGA each agent is fixed on a lattice-point and can only interact with its four neighbours, whereas the agents of GMAS may move and meet other agents in the whole environment.

1.2 Self-adaptive Genetic Algorithms

GMAS has two strong points: it uses a variable self-regulated population size and has only 2 parameters. As population size is one of the main parameters that affect the robustness and computational efficiency of the GAs [5], variable population size GAs have been proposed in literature. One of the first works dealing with this subject was proposed by Smith in 1993 [10]. The population size was adjusted based on the probability of selection error.

In 1998, Sawai and Kizu presented a Parameter-free Genetic Algorithm ($PfGA$) [8] inspired by the "Disparity Theory of Evolution". $PfGA$ applies systematic crossover and mutation operators to get rid of genetic parameters. It also features a selection process that results in a well balanced variable population size.

Koumousis and Dimou [4] demonstrated that applying a sinusoidal oscillating population size for structural problems improved the capacity of the GA to refine its solutions and reduced the sensitivity in tuning the GA parameters. Several algorithms including re-initialization phases also exist. The best known of them are certainly Eshelman's CHC algorithm [2] and Goldberg's μGA [3].

More recently, Koumousis et al. [5] proposed to make the population size alternatively increase during convergence phases and decrease during diversification/re-initialization phases. Their algorithm is named Saw-tooth GA because the variable population size follows a saw-tooth scheme with a specific amplitude and period of variation. At the beginning of each period, randomly generated individuals are appended to the population, substituting some individuals, for instance the least fitted ones. Then the population size decreases linearly until the beginning of the next period. These authors use classical test functions to compare Saw-tooth with a standard GA and with μGA [3], they conclude that Saw-tooth GA gives best results both in terms of performances and robustness. The variable population size and the re-initialization phases contained in Saw-tooth GA make it closest to GMAS than the distributed optimisation frameworks previously described. This paper provides the reader with enough details to compare his own algorithm to Saw-tooth GA. That is why this solution was chosen to assess the results provided by GMAS and presented in Sect.3.2. The following section describes GMAS main principles.

2. GENETIC MULTIAGENT SYSTEM

GMAS agents live over a 2D torus of width C (the torus is independent from the optimisation problem). There may be several agents on each node of the torus. Agent fitnesses are calculated as described in Sect. 3.3 in the case of the permanent magnet motor design optimisation. For the classical optimisation functions the fitness equals the value of the function, e.g. for a minimisation problem the algorithm minimises the fitness. GMAS maintains 3 sets of agents: a set of currently evolving agents P, a set of newborn agents B and a set of dead agents D.

GMAS main loop consists in the following steps:

1. if $|P| = 0$ then place C^2 randomly generated agents on the grid;
2. update all agents according to their fitness (best agents are updated first);
3. $P = P - D$, $D = \emptyset$;
4. $P = P \cup N$, $N = \emptyset$.

The first condition allows the algorithm to re-initialise itself during execution. Of course the best individual so far has to be stored.

An agent a is characterised by its location (x_a, y_a) on the grid, by the set of its neighbours $V(a)$ and its age (i.e. the number of iterations it has been in P). The update procedure of an agent a is as follow:

1. increment one's age;
2. update $V(a)$;
3. if $|V(a)| < T$ then do reproduction else do predation;
4. move to a random neighbour cell.

The neighbourhood $V(a)$ of an agent is the set of agents that are in the same cell and in the 8 adjacent cells. Agents update themselves sequentially but look for neighbours in both $P - D$ and N. A dead agent is no more available for next updating agents whereas newborn agents are immediately available. This brings asynchronism to the genetic process.

T is the threshold of predation. C and T are the two parameters of GMAS that are studied in Sect. 3.1.

Reproduction is similar to the reproduction in $PfGA$ [8]. A uniform multi-point crossover is applied to the agent and its best neighbour resulting in 2 children. A uniform mutation is performed on one of them. Therefore GMAS does not have explicit crossover and mutation rates. The two children are introduced into N.

Table 1: Comparison between GMAS, Saw-tooth GA, SGA and μGA

Function	GMAS	Saw-tooth	SGA	μGA
Schwefel	-4188.9	-4178.4	-4111.1	-4174.2
Griewangk	0.0011	0.087	0.092	0.229
Rastrigin	0.0027	3.02	5.50	2.85
Ackley	0.028	12.97	14.26	16.18

Agents perform predation by eliminating their worst neighbour (i.e. by putting it in D). The eliminated agent is no more available for the next updating agents, this brings elitism to the algorithm.

An evolutionary optimisation mechanism without any explicitly defined global selection process emerges from the combination of local interactions between agents. This kind of mechanism is well suited to be extended with migration between several GMAS islands.

3. PERFORMANCE EVALUATIONS

Four classical test functions where used to compare the performances of GMAS, Saw-Tooth Algorithm [8] and μGA [5], before applying GMAS to a study case of permanent magnet motor design. In order to perform experimentation in the best possible conditions, a parametric study, described in the following section, was first conducted. The hardware used for all experimentations was a dual processor Intel Xeon ($2.8\,GHz$) workstation with a $3\,GB$ memory.

3.1 Parametric Study

The parametric study is conducted with the Goldberg and Richardson function f_{gr}. The number of variables n of the function is set to 4. All of them consist in launching 20 runs of GMAS with a limit of about 5000 calls to the objective function for different values of the grid size C and the predation threshold T. C takes values between 3 (the smallest value which is coherent with the definition of a neighbourhood of size 9) and 14 (which gives an initial population of nearly 200, a value that is often used for GA populations). T takes ten values from 1 and $C \cdot C$. The study is divided in four parts. In the first part an arbitrary value of 9 is used for the predation threshold T. The best grid size in terms of mean fitness value is $C_{best_fit} = 7$ and the best grid size in terms of mean computation time is $C_{best_time} = 14$. C_{best_fit} and C_{best_time} are then used to make two studies of the predation threshold T. These two parts give best mean fitness values for $T = 11$. In the fourth part $T = 11$ is fixed. For all four parts, the best (C, T) couple in terms of mean fitness value is $(8, 11)$ and the best (C, T) couple in terms of computation time is $(14, 13)$.

3.2 Comparison with Saw-tooth GA

To evaluate the performance of GMAS, 50 runs limited to 20000 objective function calls are launched for each classical test function: Schwefel's *sine root*, Griewangk, Rastrigin and Ackley (see Table 2). For the four functions the dimension is $n = 10$. Results of the parametric study are used to fix the value of the parameters $(C, T) = (8, 11)$. Tab. 1 presents the corresponding results. It provides, for each test function, the mean of the best fitness values reached for 50 runs of GMAS. It shows that the algorithm outperforms both Saw-tooth GA and micro GA for the whole set of functions. In particular, for Rastrigin's and Ackley's functions the difference is really significant.

Fig. 1 gives an example of the evolution of several parameters or measures of GMAS during a run with the Ackley function (a mono-objective minimisation problem). The first four plots give a global image of the population of GMAS: the population size $|P|$, the number of newborns $|N|$, the number of deaths $|D|$ and the average age of the agents of P. The three last plots correspond to the mean fitness, the best fitness and the standard deviation. These plots confirm the population size is self-regulated.

3.3 Optimisation of a Permanent Magnet Synchronous Motor

The studied system is a synchronous permanent magnet electrical motor and the considered application is a driving motor of an air-circuit Fuel-Cell (FC) compressor. On the one hand, as the motor uses the electrical power provided by the FC, the required power must be less than 20 % of the global power of the FC. Consequently, maximising its efficiency is all the more important. On the other hand, given that the whole system is embedded on a vehicle, the motor weight must be as small as possible (the authors have chosen to target a weight of $2\,kg$). A complete analytical model [1], which contains 84 parameters linked by 52 equations, is used to express the efficiency $\eta(x)$ and the weight $w(x)$ as functions of the construction parameters (either discrete or continuous) gathered in the individual x. Fig. 2 illustrates some of these construction parameters. Solving the optimisation problem consists in determining the construction parameters that maximise the efficiency while maintaining the weight in the required limits. Computer-aided design software such as *Pro@DESIGN*, a tool based on a sequential quadratic programming (SQP) algorithm, already exist for such machines. However specific knowledge about the variation domain of each variable is required to use it. In some way, this determines a pre-design that is locally optimised by the SQP algorithm, moreover this kind of tool is not well adapted to the global optimisation of electrical systems. Indeed, deterministic algorithms (such as SQP) need the knowledge of the objective and constraint gradient symbolic expressions, in order to limit the divergence of the optimisation algorithm due to the numerical evaluation of the gradients. One solution consists in proposing evolutionary solving approaches that explore the search space more largely without requiring any other knowledge than the model and the objectives. These methods also present good performances and robustness. That's why many recent works focus on the design of electrical machines with the help of genetic algorithms (GAs). Sudhoff et al. [9] list various genetic approaches applied to machine design since 1997 in an electrical engineering oriented paper. In the proposed contribution the two objectives $\eta(x)$ and $w(x)$ are aggregated into a function F based on the normalised distance to both the theoretical optimal efficiency $\eta(x) = 1$ and the targeted weight $w(x) = 2$:

$$F(x) = \sqrt{|\eta(x) - 1|^2 + \frac{|w(x) - 2|^2}{w_{max} - w_{min}}}. \quad (1)$$

The objective function must also take into account several constraints related to variables which are involved in the analytical model. Obviously all geometrical parameters must be strictly positive and the RMS current density and the RMS voltage are limited by upper bounds ($5\,A \cdot mm^{-2}$ and $120\,V$). If a solution does not satisfy one or several constraints its fitness value is penalised: it is divided by 2 for each unsatisfied constraint. As in Sect. 3.2 GMAS has

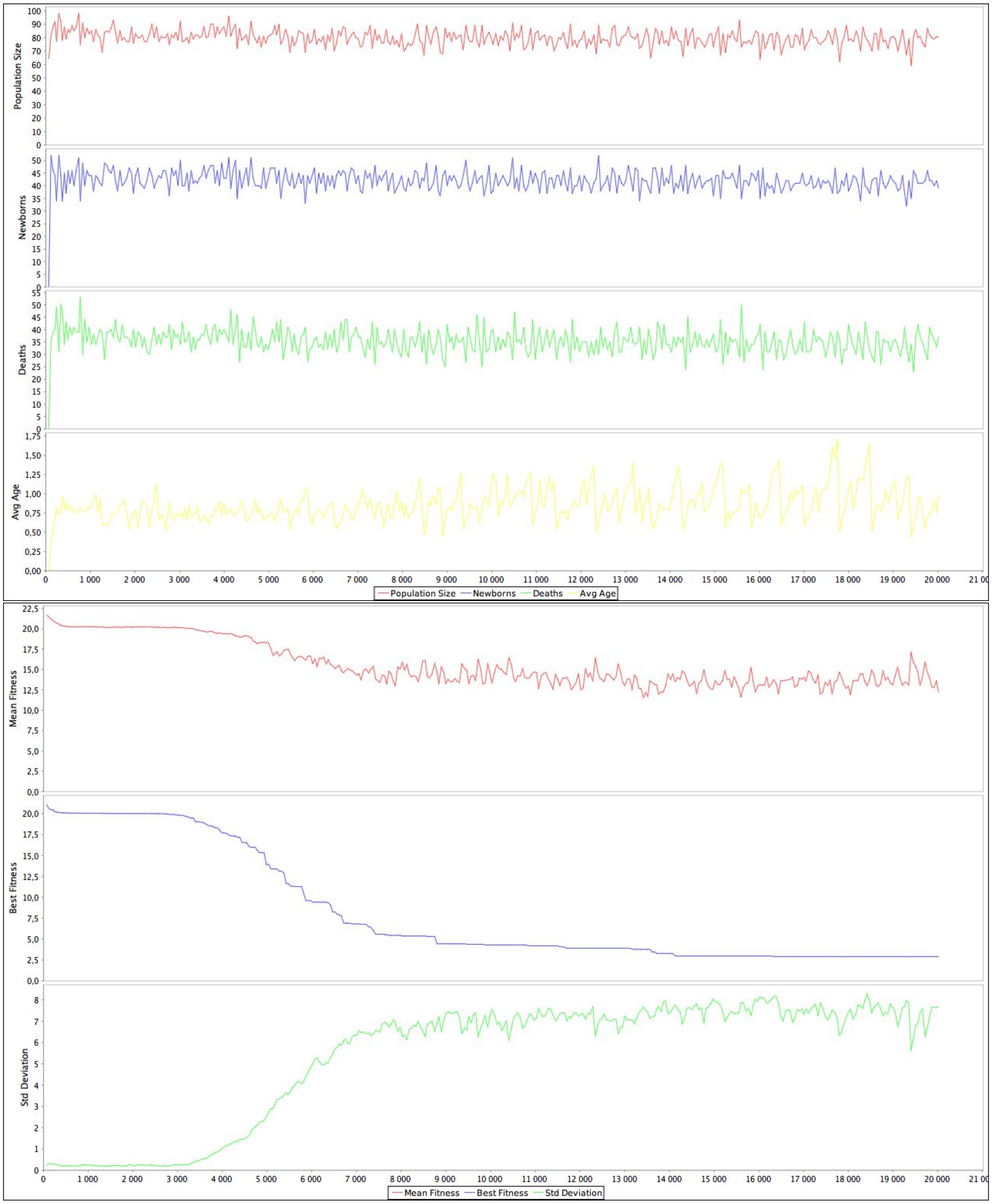

Figure 1: An example of execution for the Ackley function

Table 2: Test Functions

Goldberg and Richardson	$f_{gr}(x) = \prod_{i=1}^{n} \left[(\sin(5.1\pi x_i + 0.5))^{\alpha} \cdot \exp\left(-4.0 \cdot \log(2.0) \cdot \frac{(x_i - 0.0667)^2}{0.64} \right) \right]$
	$\forall i \in \{1, \ldots, n\}, x_i \in [0; 1]$ and $\alpha = 30$
	$f_{gr}^{max} = 1$
Ackley	$f_{ack}(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(c \cdot x_i) \right) + a + \exp(1)$
	$a = 20,\ b = 0.2,\ c = 2\pi$ et $\forall i \in \{1 \ldots n\}\ x_i \in [-100; 100]$
	$f_{ack}^{min} = 0$
Schwefel's *sine root*	$f_{sch}(x) = \sum_{i=1}^{n} \left[-x_i \cdot \sin \sqrt{\lvert x_i \rvert} \right]$
	$\forall i \in \{1 \ldots n\}\ x_i \in [-500;\ 500]$
	$f_{sch}^{min} = -4189.8$
Rastrigin	$f_{ras}(x) = \sum_{i=1}^{n} [x_i^2 - A \cdot \cos(\omega x_i) + A]$
	$\forall i \in \{1 \ldots n\}\ x_i \in [-5;\ 5]$, $A = 10$ and $\omega = 2\pi$
	$f_{ras}^{min} = 0$
Griewangk	$f_{grie}(x) = 1 + \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}} \right)$
	$\forall i \in \{1 \ldots n\}\ x_i \in [-50;\ 50]$
	$f_{grie}^{min} = 0$

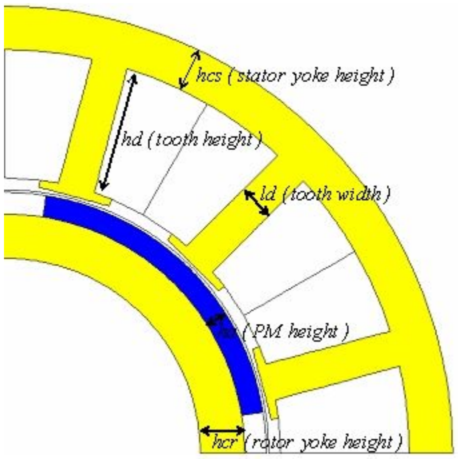

Figure 2: Geometry of the studied motor.

been launched with parameters $(C, T) = (8, 11)$ and a limit of 20000 objective function calls. Fig. 3 presents the non-dominated front obtained by a single run of GMAS. The obtained experimental results show that GMAS permits to quickly reach quite good solutions. Indeed, for this study case, providing *Pro@DESIGN* with a pre-sizing defined by an expert permits to design a motor with a weight of 2.09 kg and an efficiency of 95 %. The optimisation takes less than one second but the pre-sizing phase might take several hours. The single solution returned by SQP greatly depends on the initialisation point proposed by the expert. Solutions provided by GMAS are at least as good as the *Pro@DESIGN* solution and the mean computation time for a run of GMAS is about 70 s. GMAS requires no initialisation point and provides a great number of interesting solutions. Although the fitness function is an aggregation the non-dominated solutions may be represented by a Pareto front with respect to the two objectives as shown in Fig. 3. A great density of solutions which weight is close to 2 kg are found, but the search space exploration is not limited to this area. GMAS made it possible to design a 500 W and 10'000 rpm motor with a weight of 1.99 kg and an efficiency of 95.1 %.

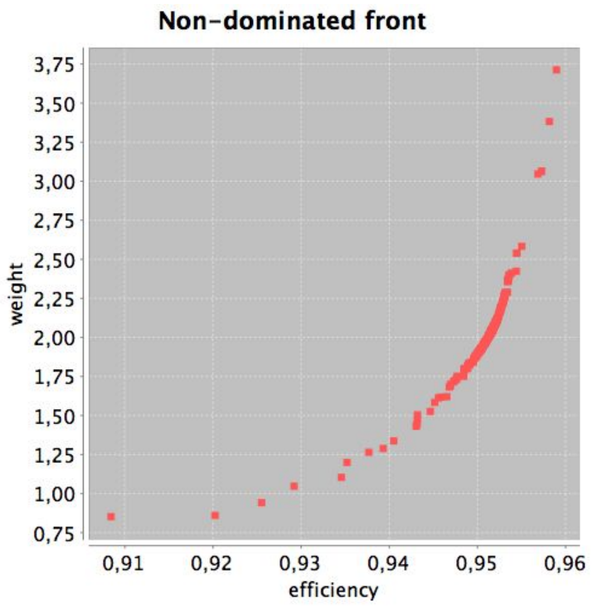

Figure 3: An example of obtained non-dominated front for the motor design problem.

4. CONCLUSION AND OUTLOOKS

This paper proposed a genetic multiagent system, GMAS, without global selection and characterised by self-adaptation. Particularly, population size, which is among the parameters that greatly affect the performances of GAs, varies according to the current state of the search, without any user interaction. The experimental results show that this algorithm outperforms Saw-tooth GA and μGA for four classical test functions. Applying this approach to design a permanent magnet synchronous motor gives as good solutions as those returned by deterministic algorithms. This approach has other advantages: no initialisation point is required, mean computation time is reasonable in comparison with the time an expert needs to determine an initial solution before using SQP algorithms, and it gives a large non-dominated front that represents a whole set of interesting alternative solutions. Finally, this work has three kinds of prospects. The first category consists in determining if a Pareto-based multi-objective version of GMAS would be more efficient. This requires an evaluation process and a local selection mechanism which are both Pareto-based. The second category focuses on the design of an island-based version of GMAS deployable on a peer-to-peer (P2P) network (i.e. a decentralised network in which heterogeneous machines share resources with each other as equals) to increase both computational power and cooperation. Deploying a instance of GMAS on each peer/island would be made easier because of its variable population size and its decentralised selection process, however efficient communication processes must be defined to ensure exchange of information between islands. Finally, the good performances and the robustness of GMAS should allow, in fine, the extension to other kinds of problem, and the integration of other various criteria, particularly environmental ones, in order to take them into account as soon as possible in the design process.

5. REFERENCES

[1] Aissa Benhamou, Christophe Espanet, Didier Chamagne, and Abdellatif Miraoui. Optimisation of a synchronous permanent magnet motor: comparison between stochastic and deterministic methods. In *OIPE proceedings*, volume 1892, pages 68–73, 2004.

[2] Larry J. Eshelman. The CHC adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In *Proceedings of the First Workshop on Foundations of Genetic Algorithms. Bloomington Campus, Indiana, USA, July 15-18 1990*, pages 265–283, 1990.

[3] David E. Goldberg. Sizing populations for serial and parallel genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pages 70–79, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.

[4] V. K. Koumousis and C. K. Dimou. The effect of oscillating population size on the performance of genetic algorithms. In *4th GRACM Congr. Computational Mechanics*, number 099, 2002.

[5] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, February 2006.

[6] Michela Milano and Andrea Roli. MAGMA: a multiagent architecture for metaheuristics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(2):925–941, 2004.

[7] Gregorio Toscano Pulido and Carlos A. Coello Coello. The micro genetic algorithm 2: Towards online adaptation in evolutionary multiobjective optimization. In *EMO*, pages 252–266, 2003.

[8] Hidefumi Sawai and Sachio Kizu. Parameter-free genetic algorithm inspired by "disparity theory of evolution". In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 702–711, London, UK, 1998. Springer-Verlag.

[9] S.D. Sudhoff, J. Cale, B. Cassimere, and M. Swinney. Genetic algorithm based design of a permanent magnet synchronous machine. In *2005 IEEE International Conference on Electric Machines and Drives*, pages 1011–1019, May 2005.

[10] R.E. Smith. Adaptively resizing populations: an algorithm and analysis. In *5th Int. Conf. Genetic Algorithms*, page 653, 1993.

[11] El-Ghazali Talbi and Vincent Bachelet. Cosearch: A parallel cooperative metaheuristic. *Journal of Mathematical Modelling and Algorithms*, 5(1):5–22, April 2006.

[12] Weicai Zhong, Jing Liu, Mingzhi Xue, and Licheng Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on Systems, Man and Cybernetics*, 34(2):1128–1141, April 2004.