A New Hybrid Distributed Double Guided Genetic Swarm Algorithm for Optimization and Constraint Reasoning: case of Max-CSPs

Asma Khadhraoui[1], Sadok Bouamama[2]

[1] SOIE Laboratory, University of Tunis,
khadhraoui.asma@yahoo.fr
Tel:(+216) 96 67 02 24

[2] SOIE Laboratory, University of Tunis,
Sadok.bouamama@ensi.rnu.tn
Tel: (+216) 98 44 71 89

Abstract

In this paper we propose a new distributed double guided hybrid algorithm combining the particle swarm optimization (PSO) with genetic algorithms (GA) to resolve maximal constraint satisfaction problems (Max-CSPs). It consists on a multi-agent approach inspired by a centralized version of hybrid algorithm called Genetical Swarm Optimization (GSO). Our approach consists of a set of evolutionary agents dynamically created and cooperating in order to find an optimal solution. Each agent executes its own hybrid algorithm and it is able to compute its own parameters. Our approach is compared to the GSO. It demonstrates its superiority. We reached these results thanks to the distribution using multi-agent systems, diversification and intensification mechanisms.

Key words

Genetic Algorithm, Particle Swarm Optimization, Max-CSPs, multi-agent system, guidance probability.

1 Introduction

Genetic algorithms (GA) and Particle Swarm Optimization (PSO) belong to the family of the evolutionary algorithms which proved their capacities to resolve successfully difficult problems in various domains. These problems include the constraint satisfaction problems CSPs. However both GA and PSO have strength and weakness. Several attempts of combination between both methods were registered in the literature to benefit from their advantages [5] [6] [9] [13].

One of the most known problems which can be however resolved by GAs and PSO is the Constraint Satisfaction Problems: CSPs. Several problems of the real life can be easily formulated in the form of CSP. A CSP can be defined as a triple (X, D, C) where:

- $X =\{X_1, X_2, \ldots, X_n\}$ is a set of n variables,
- $D =\{D_1, D_2, \ldots, D_n\}$ is a set of domains, for each variable Xi is associated a domain Di,
- $C= \{C_1, C_2, \ldots, C_n\}$ the constraints are the relations between the variables.

A CSP solution is an instantiation of the variables with values from the respective domains which satisfies all the constraints and the limitations imposed by domains.

In this work we are interested in one of CSP extensions: Max-CSPs for Maximal Constraint Satisfaction Problems. A Max-CSP is a CSP where a solution corresponds to an instantiation of all the variables which satisfies the maximum of constraints. Other extensions made the object of the research for these problems: the dynamic CSPs, the distributed CSPs, the CSOP (for Constraint Satisfaction and Optimization Problems), the ∑CSPs, etc. Seen the importance of these problems several methods were implemented for the resolution of Max-CSPs. These methods are classified in two categories: the complete and the incomplete methods. The first are able to provide an optimal solution but they are limited by the combinatorial explosion. The second family such as Guided Genetic Algorithm (GGA), Distributed Double Guided Genetic Algorithm (D^2G^2A), Dynamic Distributed Double Guided Genetic Algorithm (D^3G^2A) and Dynamic distributed Double Guided Particle Swarm Optimization (D^3GPSO) sacrifice completeness for efficiency. They have the advantage to escape from local optima.

The present work is inspired by the D^3GPSO [1] and the D^3G^2A [2]. These two algorithms are based on the multi-agent systems. They outperform respectively the centralized versions of GA and PSO. Our approach is a combination between these lasts.

This paper is organized as follow: the first section recalls the basic principles of GA and PSO, it recalls the D^3G^2A and the D^3GPSO then it presents the different hybrid algorithms between GA and PSO registered in the literature. The second section presents the Dynamic Distributed Double Genetic Swarm Optimization D^3GSO, the basic concept, the global dynamic and the agent structure. The third section details the experiments and the experimental results. Finally a conclusion and possible perspectives are proposed.

2 Genetic Algorithm, Particle Swarm Optimization and Hybrid algorithms

2.1 Genetic Algorithm

2.1.1 Basic concepts of GA

Genetic Algorithms (GA) were first introduced by John Holland in the 1960s. GAs belong to the family of the evolutionary computation techniques who are inspired by the concept of natural selection of Charles Darwin.

The terminology used in the genetic algorithms is traced on that of the theory of evolution and the genetics. The GA maintains an initial population p_i of potential solutions called *individuals* or *chromosomes*. Every chromosome is a collection of elements known under the name of *genes*. These last ones undergo mechanisms of *selection*, *crossing* and *mutation*. For an optimization problem, an individual represents a point of the search space, a potential solution. We generate in an iterative way populations of individuals to which we apply processes of *selection*, *crossing* and *mutation*. The selection aims at favoring the best elements of the population for the considered stopping criterion, the crossing and the mutation guarantees the exploration of the search space.

2.1.2 D^3G^2A for Dynamic Distributed Double Guided Genetic Algorithm

The Max-CSPs was treated by both complete and incomplete methods such us *Extended Forward Checking* [7] and the *Branch and Bound* algorithm [12]. They are limited by the combinatorial explosion. Whereas the incomplete methods are able to find optimal solutions without being caught up in local optima. The distributed Simulated Annealing is an incomplete method that was successfully applied to different domains such as static Max-CSPs, dynamic Max-CSPs, etc.

Other algorithms were also implemented for the resolution of Max-CSPs, we quote: *Double Guided Genetic Algorithm, Distributed Double Guided Genetic Algorithm.* The D^3G^2A was implemented inspiring by the law of the strong races, this aims not only to reach a strong race but also to use this principle as an optimization tool. This approach acquires its dynamic aspect from the type of agent that it uses. They are evolutionary agents able to calculate their genetic parameters before beginning the optimization process.

A CSP solution is represented by a chromosome. The main inspiration consists in partitioning the search space into sub-spaces. Each agent, called *Specie* agent, is responsible for one sub-space. A given sub-space consists of chromosomes having equal fitness values called FV. FV represents the specificity of the species agent $\text{Specie}_{FV.}$ Each specie agent executes its own Genetic algorithm. The species agents cooperate to find the optimal solution for the problem.

The D^3G^2A is double guided by *template,* guidance probability and min conflict heuristic (see section 3 for details). We also distinguish another type of agent that should be the mediator between species agents and the user, it is called *Interface* agent. It detects the best local solution reached by each specie agent, it returns the global best solution and it can also create new Species agents. The D^3G^2A was successfully applied to many extensions of the CSPs such as CSOPs, ∑CSPs, etc [2] [3] [4]. This algorithm outperforms centralized Genetic Algorithm which is known to be expensive in time.

2.2 Particle Swarm Optimization

2.2.1 Basic concepts of PSO

Particle Swarm Optimization (PSO) [10] was first introduced by Eberhart and Kennedy in 1995. The PSO conducts searches using a population of *particles* which correspond to individuals in GA. A group of particles is called swarm. We first generate randomly a population. Each particle corresponds to a potential solution and it has a position represented by a position vector $\overrightarrow{xi}$. It moves from a position to another with a *velocity* represented by a vector $\overrightarrow{Vi}$. The movement of a particle in the search space depends on its local knowledge and on knowledge of all the swarm. During each iteration the particle is accelerated toward the particle's previous best position and the global best position. The new velocity value and position value are calculated according to equations 1 and 2.

$$\overrightarrow{Vi}(t) = w\,\overrightarrow{Vi}(t) + \alpha\left(P_{lbest} - x_{id}\right) + \beta(P_{gbest} - x_{id}) \quad (1)$$

$$X_{id} = X_{id} + \overrightarrow{V}_{id} \qquad (2)$$

The search space dimension is *D*. X_{id} is the position value of the particle *i* in the dimension *d*, $\overrightarrow{Vi}(t)$ is the velocity of the particle at time ***t*****.** *w* is the inertia weight, P_{gbest} is the position of the global best particle, P_{lbest} is the particle's previous best position and α and β are random values in the range [0.0,1.0].

2.2.2 D^3GPSO for Dynamic Distributed Double Guided Particle Swarm Optimization

The D^3GPSO is a distributed PSO. It is a group of agents dynamically created and cooperating in order to solve a problem. Each agent performs locally its own PSO algorithm. It is inspired by the D^3G^2A.

This algorithm uses the same principle as the D^3G^2A, it consists on dividing the initial population into sub-populations and affecting each one to an agent. Each agent is also called *Specie* agent and is responsible of a set of particles having their fitness values in the same range. This range is called FVR (for fitness value range) and it is the specificity of the specie agent Specie_{FVR}. The species agents cooperate all by exchanging solutions to reach the optimal one. In fact each one executes its own double guided PSO algorithm. The latter is double guided by the concept of *template* and the min-conflict heuristic. It is enhanced by new parameters: guidance probability P_{guid}; a local optimum detector LOD and a weight ε (used by species agents to calculate their own PSO parameters). For the D^3GPSO we distinguish also a mediator agent to manage the communication between the species agents. This agent, called *Interface* agent, can also create new species agents if necessary.

The D^3GPSO was applied to many CSPs extensions [1] such as CSOPs and ∑CSPs. Compared to D^3G^2A and centralized PSO, it has been shown to be better in terms of CPU time and quality of solution
This efficiency is due to the use of the guidance probability and the mechanism of computing particle's position. Which is sometimes random, to diversify the search process and escape from local optima and sometimes guided to improve the solution's quality and increase the intensification. Each agent has a simple structure: its acquaintances (the agents it knows and with which it can communicate), a local knowledge composed of its static and dynamic knowledge and a mailbox where it stores the received messages to be later processed.

2.3 PSO and GA hybrid algorithms

Several attempts of hybridization between GA and PSO were registered in the literature.

Farooq, Shahzad and Zahid and [13] proposed a **hybrid GA-PSO**. It is a fuzzy system for user identification on Smart Phones. They proved that, compared to Ant Colony Optimization, PSO and GA, their approach return best results. Indeed, the average error rates drop quickly.

Chia-Feng Juang proposed a hybrid algorithm [5] between GA and PSO. This algorithm automates the design of recurrent network design. It is called **HGAPSO**. It was applied to the recurrent network design and compared to GA and PSO; it demonstrated its superiority with regard to both methods.

Recently, a new hybrid evolutionary algorithm suitable of the optimization of large-domain electromagnetic problems was published by Grimaldi, Gandelli, Grimaccia, Mussetta and Zich. The hybrid algorithm was called **Genetical Swarm Optimization** (GSO) [8]. It represents cooperation between both methods by maintaining their operations. In each iteration, the population is divided into two sub populations according to a hybridization coefficient HC and it is evolved with the two techniques respectively. At the end of the iteration the two parts are recombined in the new population which is going to undergo the same process in the next iteration. The GSO is a fast method for optimization of complex nonlinear objective functions. It is able to find an optimal global solution quickly without getting trapped in a local optimum. The GSO makes the framework of our approach.

3 D^3GSO for Dynamic Distributed Double Guided Genetic Swarm Optimization

3.1 Why?

By leaning on the key points of the genetic algorithms and those of the particle swarm optimization, several combinations of these two methods are presented in the literature. Furthermore, seen the advantages known for multi-agents systems (MAS), our approach is based on them and benefit from their advantages to mitigate the temporal complexity of the optimization process.

The aim of this work is to develop a hybrid algorithm of GA and PSO. This algorithm is a distributed approach to resolve the discrete Max-CSPs randomly generated. It is inspired by the works of Bouamama [1][2]. In fact, it is a group of agents dynamically created and cooperating to solve the problem. We distinguish two types of agent: *interface* agents and *specie* agents. Our approach uses the same agent's structure used in D^3G^2A and D^3GPSO. Firstly the *interface* agent generates randomly a population of individuals. They are randomly dispersed in the search space. Then it divides the population in parts, each part contains the set of individuals that have the fitness value in the same rang FVR, it is inspired by the work made in [3]. A Specie agent called Specie_{FVR} is responsible for this set of individuals.

The first improvement mechanism brought by our approach is the distribution by MAS what allows to reduce automatically the CPU time with regard to the centralized hybrid algorithm GSO.

The other improvement mechanism is the diversification in the search process in order to escape from local optima. The simplest mechanism to diversify the search is to consider a noise part during the process. For these reasons the mutation in the genetic process is sometimes random what diversifies the search and it is in other times guided to intensify the search. Also at the PSO process, the particle's position update is sometimes random and other times guided by the guidance probability P_{guid} [4].

3.2 Basic principles

3.2.1 Template concept and min-conflict heuristic

The *template* concept was introduced by Tsang [6]. It is made up of weights referred to as $template_{i,j}$ and noted $\delta_{i,j}$. Each one of them corresponds to a $variable_{i,j}$ where *i* refers to the individual and *j* to the position. $\delta_{i,j}$ represents the number of constraints violated by the variable j. A weight is an integer not exceeding the total number of the problem's constraints. In our approach a template is attached to each individual. For the part of the population that will execute the double guided GA, an individual is called chromosome and a template is attached to each one. For the other part that will execute the double guided PSO, each individual is called particle. And for each one is attached a template. In our case the two parameters α and β mentioned in equations 3 and 4 will be functions of templates.

$$\alpha = template_{lbest}/(template_{lbest} + template_{gbest}) \ (3)$$
$$\beta = template_{gbest}/(template_{lbest} + template_{gbest}) \ (4)$$

These equations prove that the best particles have more chance to be pursued by others. For each solution, the min-conflict heuristic determines the variable which violates most constraints. Then the agent selects the best value from this variable domain. This value has to minimize the number of violated constraints. Then the Specie agent instantiates the variable with this value.

3.2.2 Guidance probability and local optimum detector

The mutation process in the genetic process and the particle's position update in the PSO process are sometimes random what diversifies the search and they are in other times guided by a guidance probability P_{guid} to intensify the search process.

The PSO search process executes a random movement with a probability P_{guid} and will be guided and intensified with a probability 1- P_{guid}. The random movement for a particle could be done according to equation (5)

$$x_{id} = random_{[0,1]} * x_{id} \quad (5)$$

In the other hand, the local optimum detector LOD is an operator that we use in the PSO process. It represents the number of iterations in which the neighboring does not give improvement. If the best solution found by a specie agent remains unchanged for LOD generations we can conclude that the particles are blocked in a local optimum. So, the best particle having this fitness value will be penalized. This variable is given by the user at the beginning of the optimization process but it is changed by each specie agent according to the attained fitness value.

3.2.3 Agent structure

The *interface* agent is the mediator agent between all the species agents of the problem. As acquaintances it has all the species agents. The acquaintances are specified by the user in the beginning. Its static knowledge includes the CSP data and the parameters specified by the user (P_{cross}, P_{mut}, P_{guid}, LOD, ε). Its dynamic knowledge includes the best solution.

The *Specie* agent has as acquaintances the other species agents and the interface agent. Its static knowledge consists of the CSP data (the variables, their respective domains and the constraints), its specificity (the FVR value) and other parameters specified by the user. As dynamic knowledge it has the set of the individuals of its sub-population.

3.3 Global dynamic

The work is a combination between a GA and a PSO. Furthermore it is distributed and dynamic. We were inspired by behavior of the agents in the works realized by Bouamama in [3] [4]. We thus use the same structure of the species agents such as it was presented in the algorithms D^3GPSO [1] and D^3G^2A [2].

To resolve a Max-CSP, the *interface* agent generates randomly the initial population and it divides it into sub-populations according to the fitness value FV of each individual and affects the sub-population to a specie agent called *Specie*$_{FVR}$. A sub-population contains a set of individuals that have their fitness values in the same range FVR. This latter is the specificity of the agent *Species*$_{FVR}$. It is by referring to the work made by Bouamama, Ghedira and Zaeir [3]. After creation of all the species agents, the *interface* agent asks them to begin the execution of its optimization process (which is a combination between the two approaches GA and PSO).

Before starting its hybrid algorithm, each specie agent has to initialize all the templates corresponding to its individuals and it calculates its genetic operators (P_{cross} and P_{mut}). After that, it divides its sub-population into two sub-populations according to the hybridization coefficient *HC* [8]. It expresses the percentage of population that, in each iteration, is evolved with GA: so *HC* =0 means that the specie agent will execute only PSO, a *HC*=1 means that the specie agent will execute only GA. While 0<*HC*<1 means that the corresponding percentage of the population is evolved with GA and the rest with PSO algorithm. After executing each algorithm (for GA it executes successively selection – crossover – mutation and for the PSO it updates the position and velocity), for each individual, the specie agent computes the new fitness value *FV*. Two cases may occur:

- *FV* is equal to the specificity of the specie agent (in the same range): in this case the specie agent keeps the individual in its sub-population;
- *FV* is not in the same range *FVR*, in this case, the individual is sent to the corresponding specie agent if it exists otherwise it is sent to the interface agent. This latter creates a specie agent having *FV* as specificity and transmits the individual to it.

Once the specie agent is created, the *interface* agent informs all the agents and asks them to execute the optimization process. The agents pursue their execution until reaching the stopping criterion which is a generations number fixed by the user or an optimal fitness value. Once this criterion is reached, the *interface* agent collects all local optimal solutions, detects the global optimal solution and sends it to the user.

3.4 Dynamic approach

Our work consists in implementing a dynamic hybrid approach between GA and PSO. It is a multi agent approach. It acquires its dynamic aspect from the agents that it uses. Indeed, they are capable to calculate their own parameters on the basis of the parameters given by the user (P_{cross}, P_{mut}, lod, ε). $\varepsilon \in [0,1]$

Before starting its process the specie$_{FVR}$ computes P_{mutFVR} and $P_{crossFVR}$ and LOD.

If FVR > N/2 where N is the total number of constraints:

$$P_{crossFVR} \leftarrow P_{cross} * \varepsilon \quad (6)$$
$$P_{mutFVR} \leftarrow P_{mut} * \varepsilon \quad (7)$$

If FVR < N/2

$$P_{crossFVR} \leftarrow P_{cross}/\varepsilon \quad (8)$$
$$P_{mutFVR} \leftarrow P_{mut} * \varepsilon \quad (9)$$

The LOD operator is also calculated on the basis of the specificity of the specie agent [1]. The behavior of each Specie agent is presented in the Figure 1.

```
Apply_behavior(InitialPopulation)
1.  init-local-knowledge
2. while max_numb_iter do
3.     majTemplate(InitialPopulation)
4.     $P_{crossFVR}$ ← countCrossOperator($P_{Cross}$)
5.     $P_{mutFVR}$ ← countMutOperator ($P_{mut}$)
6.     $LOD_{FVR}$ ← countLOD (lod)
7.     GAindiv; PSOindiv ←  divPop(InitialPopulation,hc)
8.     GApop ← Geneticgorithm(GAindiv)
9.     PSOpop ← ParticleSwarmOptimization (PSOindiv)
10. end while
11. end
```

Figure 1 : Behavior relative to a Specie$_{FVR}$

4 Experimentations

4.1 Introduction

To validate our approach we experiment it on discrete Max-CSPs. The approach is compared with the centralized algorithm GSO [8] which proved to be better with regard to only GA and only PSO. It is also compared to D^3G^2A and D^3GPSO.

All the implementations are done with *Madkit* [11]. It is a modular and scalable multi-agent platform written in JAVA and built upon the AGR (Agent-Group-Role) organizational model. The choice of *Madkit* is justified by its simplicity of implementation, its facility to integrate the distribution in a network, its good simulation of the parallelism and the heterogeneousness of the applications and the types of agents that we can use.

4.2 Experimental design

In all the experiments we used randomly generated CSP by using the random CSP generator algorithm [4]. The latter uses CSP classical parameters: number of variables n, domain size d, constraint density p (a number between 0 and 100% indicating the ratio between the number of the problem effective constraints to the number of all possible constraints) and constraint tightness q (a number between 0 and 100% indicating the ratio between the number of pairs of values not allowed by the constraint to the size of the domain cross product).

As numerical values we used n = 20 and d = 20. And we have chosen 0.1, 0.3, 0.5, 0.7, 0.9 for the parameters *p* and *q*, we obtain 25 combinations of density-tightness. For each couple *(p,q)* we generate randomly 20 CSP examples. Consequently we have in total 500 examples. After that we have executed 35 experimentations per example and taken the average without considering outliers. As GA's parameters, all the experimentations use generations number equals to 20; a population size equals to 2000; a crossover probability P_{cross} equals to 0.5 and a mutation probability equals to 0.2 [4].
In [4] it was proved that the best value of P_{guid} is approximately 0.5. This value can be explained by the fact that not only the random mutations are important but that the guided ones are also important.

On the subject of PSO parameters we were inspired by the work made in [3], we used the PSO P_{guid} equal to 0.6, the LOD has an initial value equal to 3 and ε is equal to 0.6.

4.3 Experimental results

The performance of the two approaches will be compared as follow:

- For the same fitness value, we compute the CPU time elapsed by D^3GSO, D^3GPSO, D^3G^2A and by GSO to reach each fitness,
- For fixed CPU time, we calculate the fitness value reached by each algorithm.

According to the Figure 2, by comparing the D^3GSO with the GSO for fixed fitness values, we can notice that the first one requires slightly less time for the easy CSP examples (that have low values of *p* and *q:* the least hard). Whereas for the hard problems (having high values of density/ tightness), the improvement in CPU time with D^3GSO is important.

This improvement in the CPU time is due to three factors:

The first factor is the distribution using the multi-agents approach. In fact, the interaction between the agents in a distributed algorithm reduces its run time. The communication between the agents through the messages exchange throughout the execution helps the algorithm to find good results in short time. In fact, when a *specie* agent finds an optimal solution, it sends an immediate message to the *interface* agent. This latter does not wait for the end of the other species; it stops them and returns the optimal solution.

The second factor is the diversification by the LOD operator, the random mutations and the random update of the particle's position. This mechanism increases the convergence of the algorithm by escaping from local optima.

The third factor of improvement is the double guide which intensifies the search and helps the process to reach optimal solutions more quickly.

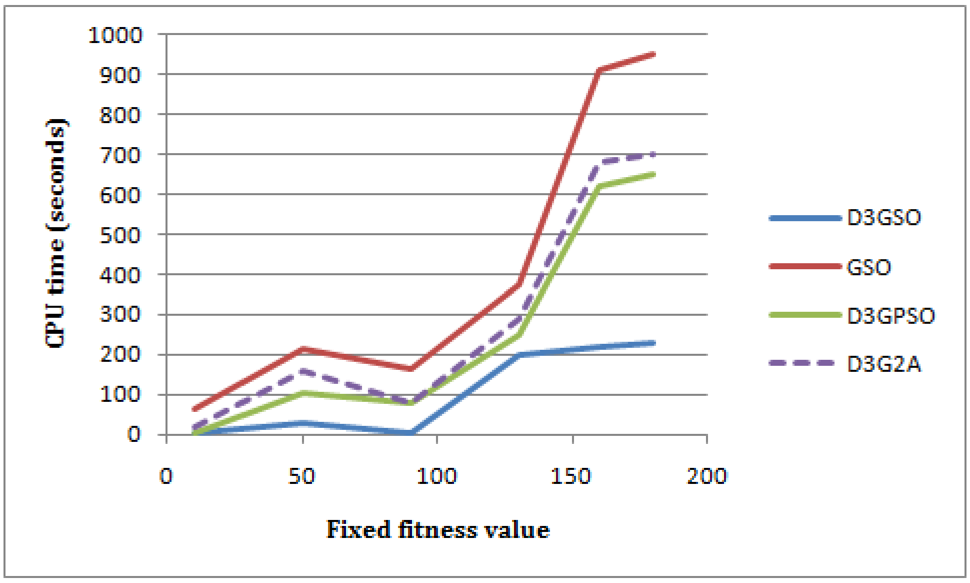

Figure 2: CPU times in seconds given for fixed fitness values

We also compare our approach to D^3GPSO and D^3G^2A. The D^3GSO performs better than D^3GPSO reaching best fitness value for the same CPU time. Also, compared to the D^3G^2A, our approach requires clearly less time for all the fitness values reached. From these results we can conclude that the hybridization between the two methods reduces the run time of the approach and reaches best solutions without getting trapped in local optima. The hybridization takes the best characteristics from the two evolutionary methods. These results confirm those found in [5] [8] and [13] which prove that combination of GA and PSO gives better results than using them separately. This is due to the simultaneous execution of the two approaches.

Form the solution quality point of view (shown in Figure 3); the D^3GSO always succeeds in finding solutions better than those found by the GSO, D^3G^2A and D^3GPSO. Indeed, for the same CPU time, it finds a solution better than that found by the other algorithms. The best results are perceived for the most difficult set of problems.

The Figure 3 shows well these results which are owed to the diversification and intensification which we have just detailed.

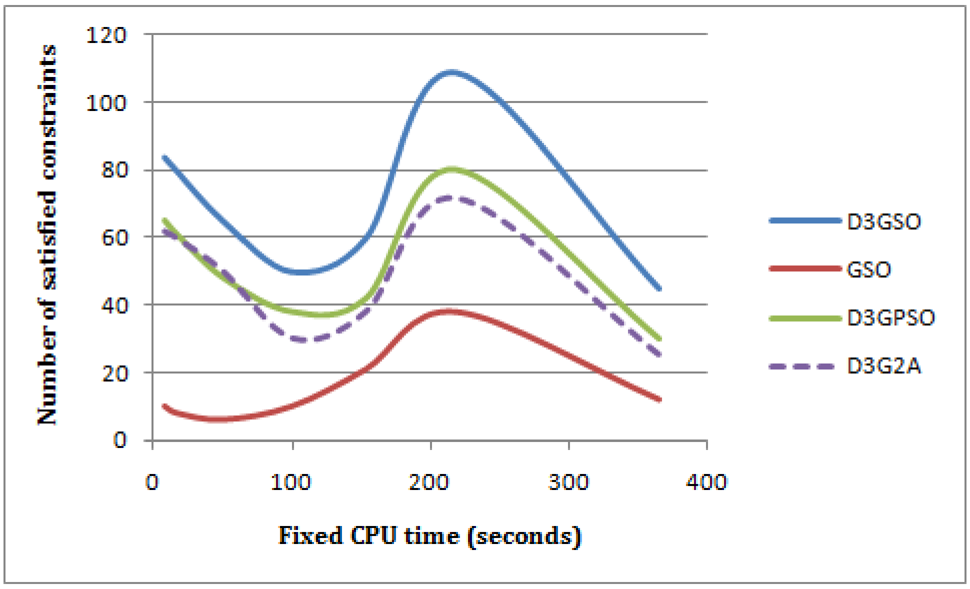

Figure 3: fitness values given for fixed CPU times

Our approach gives better solutions for hard problems. But it always remains to choose well the good value of the hybridization coefficient HC. The study made in [8] shows that the best value of this parameter is included between 0.2 and 0.4. Thus for any value belonging to this interval, we always reach good results especially in term of CPU time with regard to the separate use of GA and PSO.

We made our experiments by using at first 0.2 as value of HC afterward with 0.3 and finally with 0.4. We noticed that HC=0.3 gives better results with regard to the two other values.

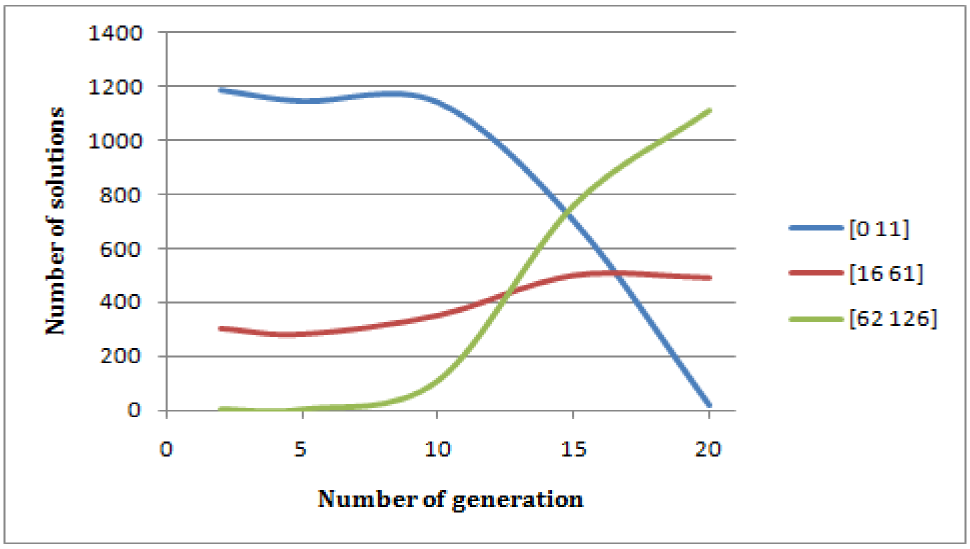

Figure 4: Evolution of solution's number of different qualities of D^3GSO

The Figure 4 presents the variation in the number of solutions with different fitness value over generations of the D^3GSO. From the shape of the curve obtained, we can clearly see that during the execution, D^3GSO moves toward better solutions. Indeed the Figure 4 shows that the number of solutions maintained by the specie agent responsible for individuals having their fitness in the range [0,11] ($\text{Specie}_{[0,11]}$) decreases rapidly to the half in the 10th generation reaching 0 in the 20th generation. Whereas the $\text{Specie}_{[62,126]}$ has the largest number of solutions at the end of the algorithm, it reaches 1180 solutions. With these results we can learn the effect of the distribution in D^3GSO. In fact, the exchange of solutions between the agents helps the algorithm to evolve towards better subspace of solutions.

5 Conclusions and perspectives

We have developed a new hybrid GA-PSO algorithm. It is a distributed double guided approach called Dynamic Distributed Double Guided Genetic Swarm Optimization and denoted D^3GSO. It is based on multi-agent system and guided by the min-conflict heuristic, the concept of template and by the guidance probability P_{guid}.

The performance of our approach is compared to that of the centralized hybrid GA-PSO called GSO, the Dynamic Distributed Double Guided Particle Swarm Optimization D^3GPSO and the Dynamic Distributed Double Guided Genetic Algorithm D^3G^2A. It was applied to discrete randomly generated Max-CSPs. The D^3GSO has been experimentally shown to be better in terms of CPU time and solution quality.

The improvement is due to three factors. The first one is the distribution using the multi-agents approach. In fact the interaction between the agents in a distributed algorithm reduces its run time. The second factor is the diversification by the LOD operator, the random mutations and the random update of the particle's position. The diversification increases the algorithm convergence by escaping from local optima. The last one is the double guidance which intensifies the search and helps the process to reach optimal solutions.

In future research we intend to apply the D^3GSO to other CSP extensions such as $\sum$CSPs, CSOPs, etc. Our approach can be extended to solve multi-criteria problems and continuous optimization problems.

References

[1] Bouamama S, *A new Distributed Particle Swarm Optimization Algorithm for Constraint Reasoning*. KES'2010, the 14th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems, Cardiff, Wales, UK, 2010.

[2] Bouamama S and Ghedira K, *A Dynamic Distributed Double Guided Genetic Algorithm for Optimization and Constraint Reasoning*. International Journal of Computational Intelligence Research; Volume 2, Issue 2 ,2006, pages 181-190.

[3] Bouamama S, Ghedira K and Zaeir N, *Load Balancing for the Dynamic Distributed Double Guided Genetic Algorithm for MAX-CSPs*. International Journal of Artificial Life Research, Vol 1, Issue 4, 2010.

[4] Bouammama S. and Ghedira K., *A Family of Distributed Double Guided Genetic Algorithm for Max_CSPs*. The International Journal of Knowledge-based and Intelligent Engineering Systems, volume10, Number 5, pp. 363-376; 2006.

[5] Chia-Feng Juang. *A hybrid of genetic algorithm and particle swarm optimization for recurrent network design*. IEEE Transactions on systems, MAN, AND CYBERNETICS Part B: CYBERNETICS, 34(2), April 2004.

[6] E.P.K Tsang, Wang C.J., Davenport A., Voudouris C., Lau T.L.: *A family of stochastic methods for Constraint Satisfaction and Optimization*, University of Essex, Colchester, UK, November 1999.

[7] E.C. Freuder and R.J. Wallace: *Partial Constraint Satisfaction*, AI 58, 21-70, 1992.

[8] F. Grimaccia M. Mussetta R. E. Zich E. Alfassio Grimaldi, A. Gandelli. *New hybrid technique for the optimization of large-domain electromagnetic problems*. IEEE TRANSACTIONS ON ANTENNAS AND PROPAGATION, 55 :781-785, 2007.

[9] H.P. Lee C. Lu et L.M. Wang X.H. Shi, Y.C. Liang. *An improved GA and a novel pso-ga-based hybrid algorithm*. Information Processing Letters, pages 255-261, 2005.

[10] Kennedy, J. and Eberhart, R. C. *Particle swarm optimization*. Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ, IEEE service center. pp. 1942-1948, 1995

[11] O. Gutknecht. *Madkit, A generic multi-agent platform, agents'00*. In 4th International Conference on Autonomous Agents, pages 78-79

[12] R.J. Wallace : *Enhancement of Branch and Bound methods for maximal constraint satisfaction problem*, AAAI, 1996

[13] S. Zahid, M. Farooq and M. Shahzad: *A hybrid ga-pso fuzzy system for user identification on smart phones*. GECCO'09, Montréal Québec, Canada, July 8-12 2009.