D^3G^2A: A DYNAMIC DISTRIBUTED DOUBLE GUIDED GENETIC ALGORITHM FOR THE CASE OF THE PROCESSORS CONFIGURATION PROBLEM

Sadok Bouamama, Khaled Ghédira
SOIE/University of Tunis
B. 204, departement of computer science
41, rue de la liberté, 2000 cité Bouchoucha. Tunisia

Keywords: Constraint satisfaction and optimization problems, multi-agent systems, genetic algorithms, Min-conflict-heuristic, guidance operator.

Abstract: Within the framework of Constraint satisfaction and optimization problem (CSOP), we introduce a new optimization distributed method based on Genetic Algorithms (GA). This method consists of agents dynamically created and cooperating in order to solve the problem. Each agent performs its own GA on its own sub-population. This GA is sometimes random and sometimes guided by both the template concept and by the Min-conflict-heuristic. In addition with guidance, our approach is based on NEO-DARWINISM theory and on the nature laws. In fact, by reference to their specificity the new algorithm will let the agents able to count their own GA parameters. In order to show D^3G^2A advantages, experimental comparison with GGA is provided by their application on the large processors configuration problem.

1 INTRODUCTION

CSP formalism (Tsang, 1993) consists of variables associated with domains and constraints involving subsets of these variables. A CSP solution is an instantiation of all variables with values from their respective domains. The instantiation must satisfy all constraints.

In the realms of CSP, the instantiation of a variable with a value from its domain is called a label. A simultaneous instantiation of a set of variables is called a compound label, which is a set of labels. A complete compound label is one that assigns values, from the respective domains, to all the variables in the CSP.

A CSOP is a CSP with an objective function f that maps every complete compound label to a numerical value. The goal is to find a complete compound label S such that f(S) gives an optimal value, and that no constraint is violated. CSOPs make up the framework to this paper.

CSOP which is NP-hard has been dealt with by complete or incomplete methods. The first ones, such as Branch and Bound (Tsang, 1993) are able to provide an optimal solution. Unfortunately, the combinatorial explosion thwarts this advantage. The second ones, such as Guided Genetic Algorithms (GGA) (Lau and Tsang, 1998) have the property to avoid the trap of local optima. They also sacrifice completeness for efficiency.

There is other distributed GAs known as Distributed Guided Genetic Algorithms. These approaches have been successfully applied to Max-CSP (Bouamama and Ghedira, 2003, 2004). Basically these distributed approaches outperform the Centralized Genetic Algorithms (CGAs) (Lau, 1998), which are especially known to be expensive in time. These approaches give good results with the Max-CSPs, in terms of both optimality and solution quality. Why not to apply the same idea for CSOPs. This is the aim of this paper. Our interest in GAs is also motivated by their proven usefulness in many hard optimization problems (DeJong, 1989)(Tsang, 1999), solving multiprocessor scheduling problems(Tsujimura, 1993) (Michael et al., 1999).

2 CSOP FORMALISM

A *constraint satisfaction and optimization problem*, or CSOP, is a quadruple (X, D, C, f); whose components are defined as follows:

– X is a finite set of variables $\{x_1, x_2, \ldots x_n\}$.

Bouamama S. and Ghédira K. (2005).
D3G2A: A DYNAMIC DISTRIBUTED DOUBLE GUIDED GENETIC ALGORITHM FOR THE CASE OF THE PROCESSORS CONFIGURATION PROBLEM.
In *Proceedings of the Second International Conference on Informatics in Control, Automation and Robotics*, pages 242-247
DOI: 10.5220/0001181102420247

- *D* is a function which maps each variable in X to its domain of possible values, of any type, and D_{xi} is used to denote the set of objects mapped from x_i *by D*. D can be considered as D = {D_{x_1}, D_{x_2}, …,D_{x_n}};
- *C* is a finite, possibly empty, set of constraints on an arbitrary subset of variables in *X*. these constraints are represented in Extension or in Intention.
- *f* an objective function which maps every instantiation to a numerical value.

3 DYNAMIC DISTRIBUTED DOUBLE GUIDED GENETIC ALGORITHM FOR CSOP

3.1 Basic principles

Our approach draws basically on the concept of both species and ecological niches. The species consists of several organisms having common characteristics whereas the ecological niche represents the task performed by a given species. Goldberg sets that the sexual differentiation based on specialization via both the building of species and the exploitation of ecological niches provides good results (Goldberg, 1989). A certain number of methods have been settled in order to favorite the building of ecological niches (Ghedira, 2002) in GAs.

So, the idea here is to partition the initial population into sub-populations and to assign each one of them to an agent called Species agent. A given sub-population consists of chromosomes having their fitness values in the same range. This range, said FVR, is called the specificity of the Species agent $Species_{FVR}$. Species agents are in interaction, in order to reach an optimal solution for the problem. For this reason, each Species agent performs its own GA. The latter is guided by both template (Tsang, 1999) concept and min-conflict heuristic (Minton, 1992). An intermediary agent is necessary between the society of Species agents and the user, essentially to detect the best partial solution reached during the dialogue between the Species. This agent, called Interface, may also possibly create new Species agents.

3.2 Min-Conflict-Heuristic and the Template Concept

Each chromosome is attached to a *template* (Tsang, 1999) that is made up of weights referred to as $template_{i,j}$. Each one of them corresponds to $gene_{i,j}$ where i refers to the chromosome and j to the position. $\delta_{i,j}$ represents the sum of costs of violated constraints by $gene_{i,j}$. These weights are updated by means of the *penalty* operator (see sub-section 3.7).

Templates will be used by GA in replacement. As we use the min-conflict-heuristic, replacement have to be elitist, i.e a chromosome is replaced by a better chromosome. For this, heavier templates genes have more probability to be replaced.

3.3 Preparing CSOP

Relationship between both genetic and CSOP formalisms is outlined as below; each chromosome (respectively gene) is equivalent to a CSOP potential solution (respectively variable). Moreover, each allele corresponds to a value.

Given an objective function *f,* we define an fitness function (FF) g which will be used by the optimization process (Lau, 1998).

$$g(ps) = f(ps) + \lambda * \Sigma(CP_i * I_i(ps)) \quad (1)$$

Where *ps* is a potential solution, λ is a parameter to the algorithm called Regularization parameter. It is a parameter that determines the proportion of contribution that penalties have in the fitness function.

CP_i is the penalty for $gene_i$ (all CP_i are initialized to 0) and I_i is an indication of whether ps satisfies all constraints or not:

$$I_i(ps) = \begin{cases} 1 \text{ if ps satisfies all constraints;} \\ 0 \text{ otherwise.} \end{cases} \quad (2)$$

Let us mention here that I_i is specific for every genei. for this, we sum over the index *i* the I_i values in order to express the contribution of every gene in the solution.

3.4 Agent Structure

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 one by one.

3.4.1 Species Agent

A Specie agent has got as acquaintances the other Specie agents and the Interface agent. Its static

knowledge consists of the CSOP data (i.e. the variables, their domains of values, the constraints and the objective function), the specificity (i.e. the fitness function range) and its local GA parameters (mutation probability, cross-over probability, number of generations, etc.). Its dynamic knowledge takes components as the population pool, which varies from one generation to another (chromosomes, population size).

3.4.2 Interface Agent

An Interface agent has as acquaintances all the Specie agents. Its static knowledge consists of the ∑CSP data. Its dynamic knowledge includes the best chromosome (i.e. the chromosome having the best fitness function value).

3.5 Global Dynamic

The Interface agent randomly generates the initial population and then partitions it into sub-populations accordingly to their specificities i.e. the fitness value range FVR. After that the former creates Species agents to which it assigns the corresponding sub-populations. Then the Interface agent asks these Species to perform their optimization processes. So, before starting its own optimization process, i.e. its own behaviour, each Specie agent, $Species_{FVR}$, initializes all templates and penalties counters corresponding to its chromosomes. After that it carries out its genetic process on its initial sub-population, i.e. the sub-population that the Interface agent has associated to it at the beginning. This process, which will be detailed in the algorithms, returns a sub-population "pop" that has been submitted to the crossing and mutating steps only once, i.e. corresponding to one generation. For each chromosome of pop, $Specie_{FVR}$ computes their fitness function values FV according to formula (1). Consequently, two cases may occur. The first one corresponds to a chromosome having an FV in the same range as its parents. In this case, the chromosome replaces one of the latter randomly chosen. In the second case, this value (FV) is not in the same range (FVR), i.e, the specificity of the corresponding $Species_{FVR}$. Then the chromosome is sent to another $Species_{FV}$ if such agent already exists, otherwise it is sent to the Interface agent. The latter creates a new agent having FV as specificity and transmits the quoted chromosome to it. Whenever a new Species agent is created, the Interface agent informs all the other agents about this creation and then asks the new Species to perform its optimization process. Note that message processing is given a priority. So, whenever an agent receives a message, it stops its behaviour, saves the context, updates its local knowledge, and restores the context before resuming its behaviour.

3.6 Guided Cross-over and Guided Mutation

Out of each pair of chromosomes, the cross-over operator produces a new child. The child inherits the best genes, i.e. the "lighter" ones, from its parents. The probability, for a parent chromosome$_i$ ($i=i_1$ or i_2), where $sum = template_{i1,j} + template_{i2,j}$ to propagate its gene$_{i,j}$ to its child chromosome is equal to $1\text{-}template_{i,j} / sum$. This confirms the fact that the "lighter" genes, i.e. having the best FV, are more likely than the other to be passed to the child.

For each one of its chromosomes selected according to the mutation probability P_{mut}, $Species_{FVR}$ uses the min-conflict-heuristic first to determine the gene (variable) involved in the worst FV, secondly to select from this gene domain the value that violates the minimal number of constraints and finally to instantiate this gene with this value. If all the Species agents did not meet any better chromosome at the end of their behaviour or they attain the stopping criterion, they successively transmit one of their randomly chosen chromosomes, linked to its specificity to the Interface agent. The latter determines and displays the best chromosome namely the one which have the best FV.

3.7 Penalty Operator and Local Optima Detector

To enhance the approach, the agents are given more autonomy and more dynamicity. In fact, we add an other GA's parameter that we call LOD for local optima detector. The latter represents the number of generation in which the neighboring does not give improvement, i.e. if the FV of the best chromosome remains unchanged for a specific number of generations; and so we can conclude that the agent optimization sub-process is trapped in a local optimum. In fact if the unchanged FV is lesser than the last stationary FV then automatically LOD have to be equal to one. Otherwise the LOD will remain unchanged i.e. LOD is a parameter to the whole optimization process and it will be dynamically updated by every agent.

Let us mention that every $Specie_{FVR}$ have to save its best FV for the next generations. This will be very useful not only in the case of stationary fitness values, but also to select the best chromosome in the species. In fact, if the best FV remain unchanged for LOD_i generation, the process will be considered as trapped in a local optimum. Thus for all the chromosomes having this FV, the related penalty counter PC_i of all its genes is incremented by one.

As we are in an optimization case, every $Specie_{AFR}$ have to send its best chromosome to the Interface Agent. The latter updates its local knowledge by this information. This must be done once after every generation. The Interface Agent will, at every attempt, compare the best chromosome he has with the best one sent by the species agents. Only those having the best FV will be maintained.

When the optimization process settles on a local optimum, the penalty of potential solution associated to this local optimum is increased. This helps the search process to escape from local optima, and drives it towards other candidate solutions. It is worth pointing out that a slight variation in the way that penalties are managed could make all the difference to the effectiveness of our approach. This is done by incrementing its penalty value by 1:

$$CP_i = CP_i + 1 \quad (3)$$

3.8 Mutation and Guidance Probability

The approach, as decribed until now, can not be considered as a classic GA. In fact, in classic GAs the mutation aims to diversify a considered population and then to avoid the population degeneration (Goldberg, 1989). In this approach, mutation operator is used improperly since it is considered as a betterment operator of the considered chromosome. However, if a gene value was inexistent in the population there is no way to obtain it by cross-over process. Thus, it is sometimes necessary to have, a random mutation in order to generate the possibly missing gene values. Our approach is a local search method. The first known improvement mechanism of local search is the diversification of the search process in order to escape from local optima (Schiex, 1995). No doubt, the simplest mechanism to diversify the search is to consider a noise part during the process. Otherwise the search process executes a random movement with probability *p* and follows the normal process with a probability 1-*p* (Schiex, 1995).

The mutating sub-process will change; for each selected chromosome following mutation probability Pmut, the mutation will be random with a probability 1-Pguid and guided with a probability Pguid. So that in the proposed mutating sub-process it's possible to destroy a given solution in order to enhance exploration.

3.9 Dynamic Approach

The main interest of the second improvement is based on the NEO-DARWINISM theory (Darwin, 1859) and on the laws of nature « *The Preservation of favoured races in the struggle for life* ». This phenomenon can be described, in nature, by an animal society in which the strongest members are luckier to be multiplied (so their crossing–over probability is high). The power of these elements allows them not to be infected by illnesses (mutation is then at a lower rate). On the contrary case the weakest limbs of these animals are frequently ill or unable to combat illnesses (mutation is frequent), usually this kind of animals can't attract females (reproduction is limited). In fact to cross-over a strong species and to give more mutation possibility for a weak species can be very worthy.

So, from now on Pcross and Pmut will be function of fitness and of a newer operator called ε. This operator is a weight having values from 0 to 1.

In the new optimization process, each species agent proceeds with its own genetic process. Indeed before starting the optimization process agents have to count their parameters Pcross and Pmut on the basis of their fitness values. For a given Species agent three cases are possible as described by the new genetic process

4 EXPERIMENTATION

4.1 Experimental design

The goal of our experimentation is to compare a distributed implementation with a centralized one of genetic algorithm enriched by both template concept and min-conflict-heuristic. The first implementation is referred to as Distributed Guided Genetic Algorithm (D^3G^2A) whereas the second one as Guided Genetic Algorithm (GGA). The implementation has been done with ACTALK (BRI, 89), a concurrent object language implemented above the Object Oriented language SMALLTALK-80. This choice of Actalk is justified by its convivial aspect, its reflexive character, the possibility of

carrying out functional programming as well as object oriented programming , and the simulation of parallelism. In our experiments we uses the case of the processor large configuration problem as described and formulated in (Lau, 1998). In our experiments we carry out 30 times the algorithms and we take the average without considering outliers. Concerning the GA parameters, all the experimentations employ a number of generations (NG) equal to 10, a size of initial population equal to 1000, a cross-over probability equal to 0,5, a mutation probability equal to 0,2, a probability of guidance equal to 0.5, LOD is equal to 3, λ equal to 10, ε is equal to 0.6. The performance is assessed by the two following measures:

- Run time: the CPU time requested for solving a problem instance,
- Satisfaction: the number of satisfied constraints.

The first one shows the complexity whereas the second recalls the quality. In order to have a quick and clear comparison of the relative performance of the two approaches. It's tempting to note that experimentation has been done and have the same tendency of graphics for these cases 8,12,16,20 and 32 processors. This is why we choose to present only the case of 16 and 32 processors.

4.2 Experimental results

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

- For the same value of fitness, we compute the time elapsed by each approch to attain it.
- For the same CPU time, we compute the fitness value attained by the two approches.

Figures 1 and 2 illustrates the CPU time point of view, and shows that the results provided by D^3G^2A are better compared to those given by AGG. We mention that the difference between two times becomes more significant when one increases the value of the fitness function . This proves that D^3G^2A requires less time to solve a given problem. We have come to these results thanks to agent interaction reducing GA temporal complexity. In fact, the comunication between agents helps them to in the solution investigation.

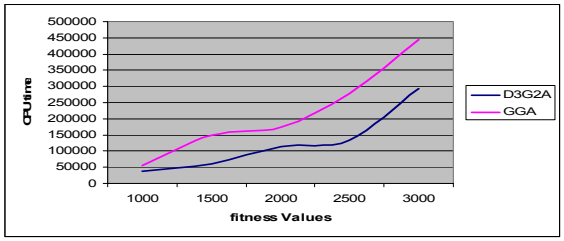

Figure 1: Generated CPU Times for fixed fitness values in the case of 16 processors

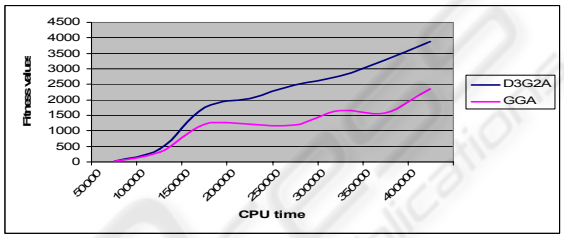

Figure 2: Generated fitness values for fixed CPU Times in the case of 16 processors

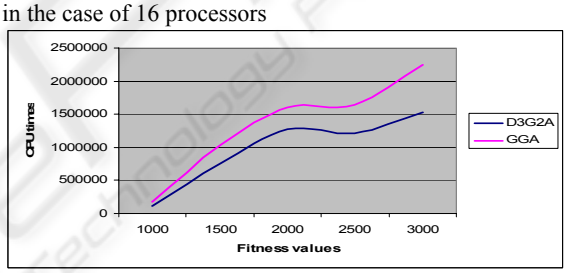

Figure 3: Generated CPU Times for fixed fitness values in the case of 32 processors

The soltion quality (the fitness function value) point of view is expressed in the figures 3 and 4. The D^3G^2A always reaches, for a fixed time CPU, a better value of the function fitness. This value is more significant for more significant times CPU. This clearly expresses the contribution in term of quality of the found solution. this is the result of the diversification and the intensification used in our approach.

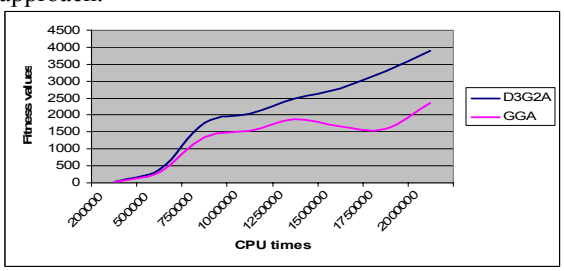

Figure 4: Generated fitness values for fixed CPU Times in the case of 32 processors

5 CONCLUSION AND PERSPECTIVES

We have developed a newer approach called D^3G^2A. This approach is a dynamic distributed double guided genetic algorithm enhanced by three new parameters called guidance probability P_{guid}, the local optima detector LOD and the weight ε,. The latter is a weight used by Species agents to determine their own genetic process parameters on the basis of their chromosomes Fitness values. Compared to the centralized guided genetic algorithm and applied to LPCP, our new approaches have been experimentally shown to be better in terms of fitness value and CPU time.

The improvement is due to both diversification and guidance. The first increases the algorithm convergence by escaping from local optima attraction basin. The latter helps the algorithm to attain optima. Consequently D^3G^2A gives more chance to the optimization process to visit all the search space. We have come to this conclusion thanks to the proposed mutation sub-process. The latter is sometimes random, aiming to diversify the search process, and sometimes guided in order to increase the best of the fitness fonction value. The genetic sub-process of D^3G^2A Species agents will no longer be the same depending on their fitness values. This operation is based on the species typology. The sub-population of a species agent can be considered as strong or weak with reference to its fitness value. For a strong species, it's better to increase cross-over probability and to decrease mutation probability. However, when dealing with a weak species, cross-over probability is decreased and mutation probability is increased. The occurrence of these measures not only diversifies the search but also explore wholly its space.

No doubt further refinement of this approach would allow its performance to be improved. Further works could be focused on applying these approaches to solve real hard CSOPs like the radio link frequency allocation problem

REFERENCES

Bouamama S, Ghédira K., 2004. ED^3G^2A: an enhanced Version of the dynamic Distributed double Guided Genetic Algorithms for Max_CSPs. *In SCI'048th World Multiconference on Systemics, Cybernetics and Informatics*. IIIS press.

Bouamama S, Ghédira K.,2003. D^2G^2A: a Distributed double Guided Genetic Algorithm for Max_CSPs. *In KES'03 the 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems, Oxford, UK*.

Darwin C.1859, *The Origin of Species*, Sixth London Editions, 1999.

DeJong K. A. , 1989. Spears W. M., Using Genetic Algorithms to solve NP-Complete problems. George Mason University, Fairfax, VA.

Freuder E.C, Wallace R.J,1992. Partial Constraint Satisfaction, Artificial Intelligence, vol. 58, p 21-70,.

Ghédira K & Jlifi B, 2002. A Distributed Guided Genetic Algorithm for Max_CSPs. *Journal of sciences and technologies of information (RSTI), journal of artificial intelligence series (RIA), volume 16 N°3/2002*

Goldberg D.E.,1989. *Genetic algorithms in search, Optimization, and Machine Learning*, Reading, Mass, Addison-Wesley, 1989.

Holland J., 1975. Adaptation in natural and artificial systems, *Ann Arbor*: The University of Michigan Press.

Lau, T. L. et Tsang, EPK., 1998. Solving the processor configuration problem with the guided genetic algorithm. *In 10ème IEEE international conference on Artificial Intelligence, Taiwan.*

Michael B., Frank M., Yi P.,1999. Improved Multiprocessor Task scheduling Using Genetic Algorithms. *In FLAIRS' 99 twelfth International Florida AI Research Society Conference*, AAAI press, p. 140-146.

Minton S. 1992. Minimizing conflicts a heuristic repair method for constraint satisfaction and scheduling problems. *In Artificial Intelligence, volume 58 pages 161-205.*

Schiex T., Fargier H. & Verfaillie G., 1995 Valued constrained satisfaction problems: hard and easy problems. *In 14th IJCAI*, Montreal, Canada august 1995.

Tsang EPK, 1993. *Foundations of Constraints Satisfaction*. Academic Press Limited,

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

Tsujimura Y., Gen M., 1997. Genetic algorithms for solving Multiprocessor Scheduling Problems. *In the 1st Asia-Pacific Conference on Simulated Evolution and Learning*, LNAI, November 9-12, Springer, vol. 1285, p. 106-115.