© 2013 Springer. Personal use of this material is permitted. Permission from Springer must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

A Hybrid System of Nature Inspired Metaheuristics in Dynamic Environments

J. M. Cadenas, M. C. Garrido, and E. Muñoz

Dpto. Ingeniería de la Información y las Comunicaciones. Facultad de Informática. Universidad de Murcia 30100-Espinardo. Murcia. Spain.
jcadenas@um.es carmengarrido@um.es enriquemuba@dif.um.es

Summary. Dynamic Optimization Problems are focusing researchers attention more and more, since they provide new challenges as well as are more similar to real problems than Static Optimization Problems. In this paper we propose to use a cooperative system of nature inspired metaheuristics to cope with them, obtaining promising results from its application to a well known test problem as is Dynamic 0-1 Knapsack Problem.

1 Introduction

Optimization problems have focused researchers interest for a long time, however this interest has mainly centered on static problems, i.e. those problems where the search space remains invariable during search time. Nevertheless, real optimization problems are rarely static, but they change over time, e.g. searching the best route between two points has to take into account traffic conditions which may vary over time, or the control of a green house [14], where the environmental conditions change over time. For this reason dynamic optimization problems (DOP) have arouse an increasing interest.

Several algorithms have been proposed to solve DOPs, being nature inspired strategies among the most successful. In fact, evolutionary algorithms (EAs) are one of the most used [3]. However its major problem is the convergence of all the population to a unique solution. This makes difficult for EAs to track the optimum of the problem as the problem changes, and many papers try to cope with this problem increasing diversity among population using various techniques [7, 14, 15]. Nonetheless EAs are not the only approaches, and we can find ant colonies, as [2], or particle swarm optimization algorithms as [8] among other metaheuristics. However another interesting paradigm as is the field of cooperative strategies has not been thoroughly studied for this kind of problems, only a few papers as [4] have appeared that use this approach.

Cooperative strategies using nature inspired metaheuristics have been applied to static problems obtaining really good results [1, 10] and showing an improvement both in robustness and quality of the solutions. In this paper we use a cooperative strategy based on a multiagent system, where two simple and different nature inspired metaheuristics (a Genetic Algorithm and an Ant Colony) controlled by a coordinator are used to solve a DOP. The use of different metaheuristics will allow the system to better track the changing optimums for two main reasons, first using various metaheuristics increments the diversity of the solutions, and second the use of different metaheuristics with different search strategies favors that at least one of them can track the optimum and guide the others. The rest of the paper is divided as follows, in section 2 the model of the Cooperative System is described, in section 3 we carry out some tests in order to test the efficiency of the strategy and in section 4 we expose the conclusions we have drawn and the works we want to develop.

2 The Cooperative System

In this section we will briefly describe the system, for a more complete description the interested reader may refer to [6] where a similar system is presented to cope with the static knapsack problem.

The hybrid system is based on a multi-agent system in which each agent implements one metaheuristic that tries to solve the problem while coordinates with the other ones. There is also a coordinating agent in charge of controlling and modifying the behavior of the meta-heuristics, thus we have a centralized model. The coordinating agent has two fundamental tasks: to collect the information on the performance of each of the solver agents and to send them orders which will affect their search behavior, [12]. The idea can be seen on fig. 1.

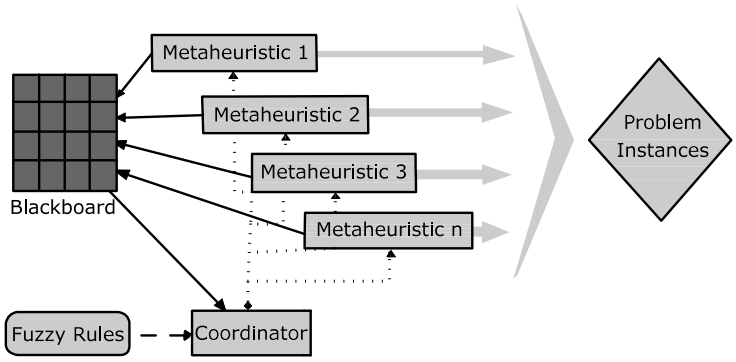

Fig. 1. Multi-agent meta-heuristic system

The communication between the agents is performed using a blackboard architecture, where each agent has an assigned space and periodically leaves information about the solution it has found to a problem. The coordinator observes this information and directs the search of each agent.

In order to give intelligence to the coordinator so that it can correctly control the interaction among the other agents, we use a template of fuzzy rules that will be completed as a result of the methodology proposed on [5]. In this methodology, we apply a supervised knowledge extraction process and as a result we obtain the information to fill the template. Once this process has been applied the rules are implemented and we obtain a completely operative system, without the need of applying the knowledge extraction process again.

This template is composed of the following rule which is replicated once for each metaheuristic:

- IF $[(wm_1 * d_1$ OR ... OR $wm_n * d_n)$ IS *enough*] THEN change the current solution of MH.

where:

- n is the number of meta-heuristics.
- MH is the meta-heuristic being evaluated by the rule.
- $d_i = (perf_i - perf_{MH})/maximum(perf_i, perf_{MH})$, where $perf$ is a measure of performance previously defined.
- $wm_i \in [0,1]$ where $\sum_{i=1}^{n} wm_i = 1$ and wm_i represents the weight of meta-heuristic i (importance of meta-heuristic i for solving the current instance).
- *Enough* is a fuzzy set with trapezoidal membership function with support contained in $[0,1]$ defined by a cuadruplet (a,b,c,d). The mathematical representation is shown in equation (1):

$$\mu(x,a,b,c,d) = \begin{cases} 0 & x \leq a \text{ or } x \geq d \\ (x-a)/(b-a) & x \in (a,b] \\ (d-x)/(d-c) & x \in [c,d) \\ 1 & x \in [b,c] \end{cases} \quad (1)$$

This rule changes the position in the search space of a meta-heuristic which is showing a bad performance for a position near the position of another meta-heuristic with a better behavior. Besides, this template also has a set of initialization rules that, depending on the instance being solved, choose the best parameters values for each metaheuristic.

Both the wm_i and the parameters values of each metaheuristic are decided using a knowledge extraction process, fig. 2, which is divided in two main phases: Data Preparation and Data Mining. In Data Preparation phase we seek to obtain a database which will hold useful information to which we can apply the necessary data mining techniques to obtain a model, this information is obtained through the resolution of a set of training instances of the problem using each metaheuristic with different combinations of parameter values. Certain data of interest are extracted from these executions.

After that in Data Mining phase we use data mining techniques to obtain the information to complete the template of fuzzy rules.

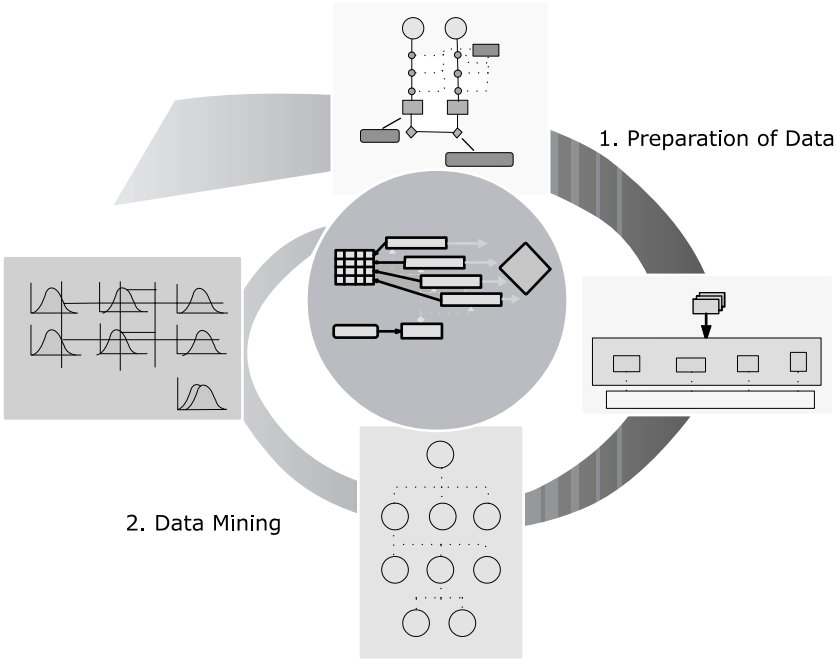

Fig. 2. Knowledge extraction process

2.1 The System Obtained

The system that we will use to cope with DOPs is composed of two nature-inspired metaheuristics, a Genetic Algorithm and an Ant Colony. After applying the process previously explained the model was obtained, with the best parameters values for each metaheuristic and with the wm_i which are 0.79 for the Ant Colony and 0.21 for the Genetic Algorithm.

Finally the system was implemented synchronously, that is, every communication is carried out at a given moment, previously specified. At this moment each metaheuristic writes its current solution and the coordinator checks which actions have to be performed. It is important to highlight the difference between the learning phase and the execution phase. In the learning phase we obtain the model of the system coordinator, being performed only once. On the other hand, in the execution phase we apply the system, previously modeled, to solve instances of the problem, without the need of applying knowledge extraction once again.

In order to obtain the fuzzy engine used to model the coordinator we used the tool XFuzzy 3.0 [11]. With this tool we modeled the different rules and obtained a fuzzy engine that was finally slightly modified to obtain an appropriate engine to our purpose.

3 Experimental analysis

In this section we want to test the efficiency of the system when facing DOPs, for that reason we will use a well known DOP as test problem, the dynamic 0-1 knapsack problem. This problem has been thoroughly studied in its static versions, and is one of the most typical DOPs.

3.1 The Dynamic 0-1 Knapsack Problem

The 0-1 Knapsack Problem (01KP) is an NP-complete problem and one of the classic problems of dynamic optimization, we have chosen it as testbed of our proposal because of this. It can be formally defined as follows: Given a set of items, each with a cost and a benefit, determine a subset such that the total cost is less than a given limit and the total benefit is as large as possible. As a mathematical formalization, it is:

$$\begin{array}{l} Max \sum_{i=1}^{n} p_i \times x_i \\ s.t. \\ \qquad \sum_{i=1}^{n} w_i \times x_i \ \leq \ C \\ \qquad x_i \in \{0,1\}\, i = 1, ..., n \end{array}$$

where

- n is the number of items.
- x_i indicates whether item i is included in the knapsack or not.
- p_i is the benefit associated to item i.
- $w_i \in [0, \ldots, r]$ is the weight of item i.
- C is the capacity of the knapsack.

It is assumed that $w_i < C,\ \forall i$ and $\sum_{i=1}^{n} w_i > C$.

In its most common version, described in [9], the Dynamic 01KP changes over time C. Our tests shall be performed with this version.

3.2 Methodology

In order to perform the test we decided that each strategy will be executed during 200 seconds, changing the instance after 10 seconds. In the case of the cooperative system, it will stop each 250 milliseconds to perform the coordination. Each instance was solved 10 times, and the results show the averages. Both the system and the independent metaheuristics were executed on an Intel core2 Quad 1.66Ghz with 2GB of Memory.

3.3 Results

In order to perform the first tests we will use the instance shown in table 1, with C changing between 60 and 104. This instance was previously defined in [9] and has been extensively used in bibliography.

Table 1. Instance of the 0-1 Knapsack Problem

Object Number	Object Value	Object Weight
1	2	12
2	3	5
3	9	20
4	2	1
5	4	5
6	4	3
7	2	10
8	7	6
9	8	8
10	10	7
11	3	4
12	6	12
13	5	3
14	5	3
15	7	20
16	8	1
17	6	2

In this tests we will compare our system with the individual metaheuristics that compose it. The results are shown in fig. 3 and fig. 4. Fig. 3 shows a comparison between the system and the Genetic Algorithm, where the value of the best solution found to the moment is shown, in it we can see how both find the same best solution for the one with higher value, on the other hand the genetic algorithm gets lost in its search of the other one while the system easily keeps track of the solution. In Fig. 4 the same comparison is shown but between the coordinated system and the Ant Colony. In this case both algorithms keep track of both solutions, however is the Coordinated System the one which obtains the best result.

As we can see, the Coordinated System always gets solutions at least equal to the best independent algorithm, and that is important because this is the minimum demanded from the system. But we can also observe how in most of the cases the system also obtains an improvement.

However, this instance seems to be very simple, for that reason we will also test our system with more complex instances as the ones that Pisinger proposes in [13]. To carry out these tests we solved a database of instances composed of 15 instances where changed both size (number of objects) and the way they were generated. Regarding size we can find three different sizes:

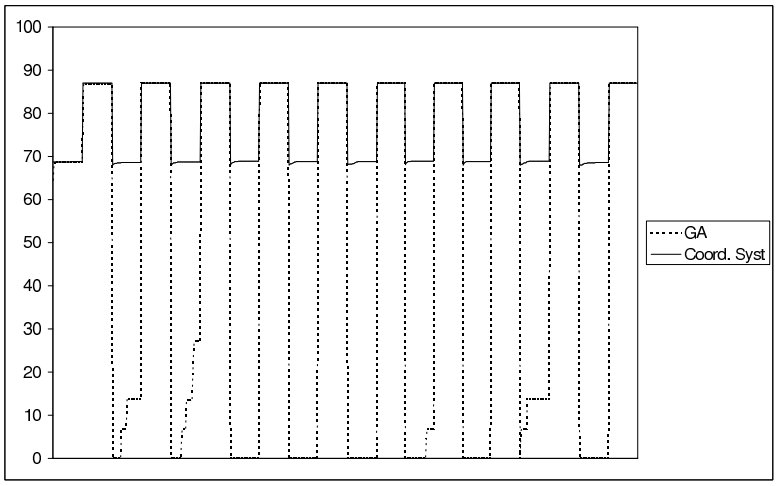

Fig. 3. Comparison between the GA and the System

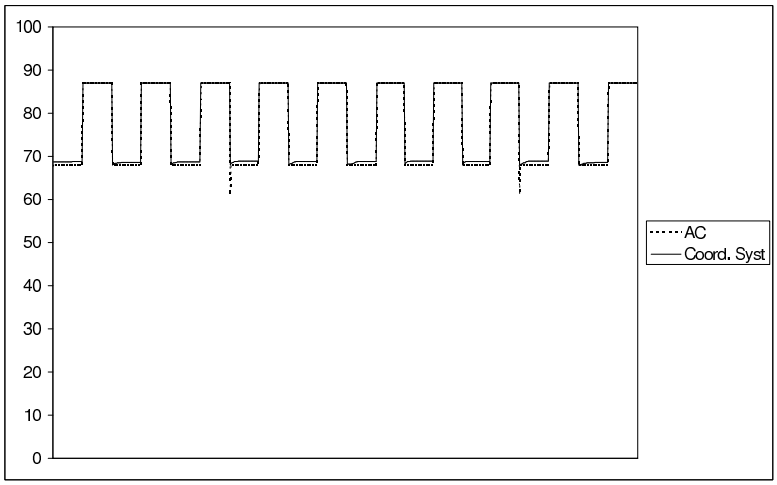

Fig. 4. Comparison between the AC and the System

500, 1000 and 1500 objects. As for the way the instances are generated there exist five types:

- Spanner: These instances are constructed in such a way that all their items are multiple of a small set of items called key. That key was generated using three distributions:
  - Uncorrelated,
  - Weakly correlated,
  - Strongly correlated.

- Profit ceiling: In these instances all the benefits are multiple of a given parameter d.
- Circle: These instances are generated in such a way that the benefits are a function of the weights, having its graph an elliptic representation.

As a result of these tests we should present as many figures as test instances, due to space restrictions instead we will show them using a quite accepted measure of performance for DOPs, the off-line error [3], which is the running average of the difference between the optimum values and the best solution encountered at any time:

$$offline\ error = \frac{1}{T}\sum_{t=1}^{T}(optimum - bestSolution)$$

Table 2. Average error of the prototype and the individual metaheuristics

Type	**AC**	**GA**	**Coord Syst**
unc span	9,2%	55,0%	0,92%
wea span	3,46%	51,1%	0,25%
str span	9,38%	52,75%	1,11%
pceil	5,28%	50,14%	0,95%
circle	13,97%	55,43%	6,03%
Items			
500	2,66%	51,13%	1,03%
1000	7,46%	52,69%	2,1%
1500	14,65%	54,83%	2,43%

Results can be seen on table 2, that shows the error percentage obtained in the resolution of the different types of instances by both the prototype and the independent metaheuristics. In the first part of the table the average error ratio is shown for each type of instance, and in the second for each instance size. As we can see the coordinated system keeps on having a better performance than the individual metaheuristics, but in contrast to the previous case where the differences where not too large, in this case, as the difficulty of the instances has increased, the differences have become more significant, obtaining in some cases an error more than ten times smaller than the best metaheuristic. It has also to be noticed that the Genetic Algorithm obtains always an error above 50% because it is only able to track one optimum, however, this fact does not affect the behaviour of the coordinated system, which has shown to be robust.

4 Conclusions and Future Work

In this contribution we propose to use a cooperative system of nature-inspired metaheuristics to cope with Dynamic Optimization Problems.

This cooperative system is composed of a Genetic Algorithm and an Ant Colony whose search is directed by a Coordinator. To model the coordinator a knowledge extraction process has been used, whose result is a set of fuzzy rules to control the exchange of solutions and which parameters values have to be used by each metaheuristic. The use of different metaheuristics allow the system to better track the changing optimums for two main reasons, first using various metaheuristics increments the diversity of the solutions, and second the use of different metaheuristics with different search strategies favors that at least one of them can track the optimum and guide the others.

The experimental results clearly showed that the proposed system is robust and effective, as it always obtained at least the same that the individual strategies that compose it, and in most cases better ones.

Regarding future work we want to test this system with more complex DOPs as for example Moving Peaks [3]. Following this line, for complex problems the cost of the knowledge extraction process may be large. For that reason we propose a different approach where a reinforcement learning process is applied. That way we will have an initial system with a "bad" performance that will be improving as it solves more instances. This system also has the advantage of obtaining the information directly from the execution, what we think can improve the performance of the system.

References

1. Alba E, Luque G, Luna F (2007). Parallel Metaheuristics for Workforce Planning. Journal of Mathematical Modelling and Algorithms. 6(3):5009–528.
2. Angus D, Hendtlass T (2005). Dynamic Ant Colony Optimisation. Applied Intelligence. 23:33–38.
3. Branke J (2002). Evolutionary Optimization in Dynamic Environments. Kluwer Academic Pub.
4. Cruz C, Pelta D, Verdegay JL (2008). A fuzzy-set based strategy in dynamic environments. Proceedings of IPMU'08. 1239–1245.
5. Cadenas JM, Garrido MC, Hernández LD, Muñoz E (2006). Towards a definition of a Data Mining process based on Fuzzy Sets for Cooperative Metaheuristic systems. Proceedings of IPMU2006, 2828–2835.
6. Cadenas JM, Garrido MC, Muñoz E (2007). A Hybrid System of Nature Inspired Metaheuristics. Nature Inspired Cooperative Strategies for Optimization (NICSO 2007) 95–104.
7. Dasgupta D, McGregor D (1992). Non Stationary Function Optimization using the Structured Genetic Algorithm. Proceedings of Parallel Problem Solving From Nature (PPNS-2) Conference. 145–154.
8. Du W, Li B (2008) Multi-strategy ensemble particle swarm optimization for dynamic optimization. Information Sciences. 178:3096–3109.
9. Goldberg DE, Smith RE (1987). Nonstationary function optimization using genetic algorithms with dominance and diploidy. Proceedings of the Second International Conference on Genetic Algorithms, 59–68.

10. Le Bouthillier A, Crainic TG (2003) A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers and Operations Research 32(7):1685–1708.
11. Moreno-Velo F.J, Baturone I, Snchez-Solano S, Barriga A (2001) XFUZZY 3.0: A Development Environment for Fuzzy Systems. International Conference in Fuzzy Logic and Technology, 93–96.
12. Pelta D, Cruz C, Sancho-Royo A, Verdegay J.L (2006) Using memory and fuzzy rules in a cooperative multi-thread strategy for optimization. Information Sciences 176(13):1849–1868
13. Pisinger D (2005), Where are the hard knapsack problems?. Computer and Operations Research 32(9):2271–2284.
14. Ursem RK, Filipic B, Krink T (2002). Exploring the performance of an evolutionary algorithm for greenhouse control. Journal of computing and information technology. 10(3):195-201.
15. Yang S, Tinós R (2007). A Hybrid Immigrants Scheme for Genetic Algorithms in Dynamic Environments. International Journal of Automation and Computing. 4(3):243–254.