Impact of Fuzzy Logic in the Cooperation of Metaheuristics

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

Dept. 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. Algorithm selection problem is a common problem when we solve optimization problems. To cope with it we have proposed a hybrid system of metaheuristics that intelligently combines different strategies using a coordinator based on Fuzzy Logic. In this paper we study the impact of Fuzzy Logic in the behaviour of this hybrid system. In order to do that we perform some test to study the impact of an important parameter, the $\alpha - cut$ used in the fuzzy engine of the system, demonstrating how the variations on this parameter may change the performance of the system with different kind of instances.

Keywords: Meta-heuristic, Cooperative System, Fuzzy Rules, Data Mining.

1 Introduction

Optimization problems have focused the interest of the research community for a long time. For that reason a large amount of strategies have been developed in order to solve them in a reasonable period of time finding solutions with a near optimum quality. However, when we try to solve different instances of an optimization problem we can find algorithm selection problem [14], which tries to decide which algorithm has to be used to solve an instance of a problem, trying to maximize a measure of performance. This problem is undecidable [7], and the most common approach to solve it is to measure the performance of a set of algorithms over a set of instances and use the one with the best performance. Nevertheless, this approach seems to be too rigid and it would be more interesting to search for more tolerant strategies that adapt better to the changing conditions of the problems.

An interesting way of obtaining flexible mechanisms is using hybrid systems, which allow us to solve complex problems, very hard to solve using less tolerant approaches. Consequently, combining intelligently different strategies using a hybrid system we can tackle the algorithm selection problem. But to obtain an "intelligent" combination of strategies that achieves good results

for all type of instances and problems we need a tolerant approach, as the one provided by "Soft Computing", specially by Fuzzy Logic. On the other hand this increase in tolerance may produce some precision loss, but we can sacrifice it in order to obtain a more robust system which could face the changes of the problems and instances. The use of different strategies together with the methodologies provided by Fuzzy Logic for building hybrid systems, will give us reasoning mechanisms and search methods which will allow us to combine domain knowledge with experimental data in order to obtain new computation tools to solve complex problems that are very difficult to solve with less tolerant approaches. In [9], Kuncheva shows a study and comparative of combination methods of fuzzy and odd nonfuzzy classifiers. The work demonstrates that better results are obtained with fuzzy combination methods and, though Kuncheva does not want to establish that the fuzzy methods are better in general, she wants to clarify that the fuzzy alternatives must not be forgotten.

In this paper, we study the impact of Fuzzy Logic in the behaviour of a hybrid system developed with the aim of solving optimization problems. In section 2, we present the design of this hybrid system, which is based on a Data Mining and Knowledge Discovery. As we use Fuzzy Logic to model some components of the system, in section 3 we show the impact of these components in the improvement of the found solutions. To finish, in section 4 we present the conclusions and work to develop.

2 A Cooperative Meta-heuristic System

2.1 Related Works

Several studies have shown that heuristics and meta-heuristics are successful tools for providing reasonably good solutions (excellent in some cases) using a moderate number of resources. An interesting trend in this area is to obtain hybrid strategies which cooperate in a parallel way in order to solve a problem, and two fields that follow this approach are *parallel meta-heuristics* and *hybrid meta-heuristics*, where the same or different metaheuristics are parallelized in order to reduce its execution time or even improve its results.

Many efforts have been focused on these fields, and we can find different implementations. First appeared synchronous implementations, where information is shared in regular intervals, such as [11]. More recently asynchronous implementations showed up, such as [6], and, according to the reports provided in [5], they obtain better results than synchronous. It has been pointed out that these approaches obtain better results than independent methods, but previous studies, [6], show that if the access to shared information is not restricted they can experiment premature convergence problems. This seems to be owe to the stabilization of the shared information produced as a result of the intense exchange of the best solutions. Trying to cope with this problem

in [13] is proposed a cooperative strategy that uses memory to control this effect. Here a coordinating agent, modeled by a set of fuzzy rules defined by the user, monitors a set of solver agents and sends orders to them about how they have to continue, implementing each agent the same meta-heuristic.

It has also been noticed that those strategies based on an unique meta-heuristic does not cover all the possibilities, thus we find two interesting challenges:

- To find ways of controlling the information exchange.
- To combine different meta-heuristics.

In this paper we face both of them and propose a similar structure to the one in [13] where a coordinator modeled by a set of fuzzy rules gets information about the performance of the different meta-heuristics and sends orders to them. The main differences are that it combines a set of different meta-heuristics and that the rules are obtained as the result of a knowledge extraction process.

2.2 Design of the System

There exist many ways to carry out the cooperative execution of different meta-heuristics. The one that seems to be more appropriate to us is a multi-agent system, in which each meta-heuristic is an agent that has to solve the problem while coordinates itself with the rest of meta-heuristics. In this schema, an approach to perform the cooperation is the use of a coordinating agent which will control and modify the behaviour of the agents, [2]. The coordinator will have two fundamental tasks: to gather information on the performance of each of the meta-heuristics and to send orders to modify their search behaviour, [2].

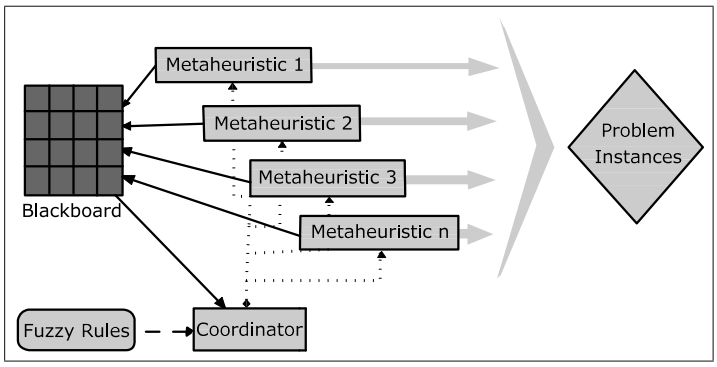

Fig. 1. Multi-agent Meta-heuristic System

To perform the communication among the different meta-heuristics we will use an adapted blackboard model. In this model each agent controls a part of

the blackboard where periodically writes the best solution it has found. The coordinator then, consults the blackboard in order to monitor the behaviour of each meta-heuristic, and decides which actions have to be taken to improve the performance.

The problem that arises immediately is how to describe the coordinator, in such a way that it can modify the behaviour of the meta-heuristics efficiently and soften the problems showed before, related with the behaviour of the meta-heuristics when taken individually.

We propose to give intelligence to the coordinator using a set of fuzzy rules, since they allow to represent data in a way very similar to human reasoning, and that will do more comprehensible the system and besides will allow to easily incorporate expert knowledge into the system as ad hoc rules. These rules will give intelligence to the coordinator and this will arise as a result of the methodology proposed on [1]. In this methodology, first, we apply a knowledge extraction process from which we will obtain the set of fuzzy rules. After that, the rules are implemented and constitute the coordinator of a completely operative system.

The knowledge extraction process is supervised and divided in three phases. It starts with the Data Preparation phase, where a database which contains useful information for data mining is obtained. Next, Data Mining phase is applied, in which the model of the coordinator of the system is obtained using data mining techniques. And the process ends with the Model Evaluation phase, where the efficiency of the set of fuzzy rules that model the coordinator is tested.

2.3 Obtaining a Set of Fuzzy Rules for the Coordinator

In this section we briefly describe the knowledge extraction process. Its details are shown in [1].

In the first phase, Data Preparation, the first thing that has to be done is select the meta-heuristics that are going to be used. With them, a broad set of training instances has to be solved, using different configurations of parameters and gathering any interesting information, including data about the instance being solved, the parameters of each meta-heuristic, the final solution and the intermediate ones. With this information a database is created and is advisable the use of a preprocess before starting with the next phase, Data Mining.

In Data Mining phase, a data mining technique has to be chosen and applied to different subsets of the database in order to fill a template of fuzzy rules, which is shown later.

Model of the coordinator

The fuzzy rule template is composed of two high level rules:

- IF $[(weight_1 * d_1$ OR ... OR $weight_n * d_n)$ IS $enough]$ THEN change the current solution of the worst meta-heuristic.
- IF $[(weight_1 * d_1$ OR ... OR $weight_n * d_n)$ IS $high$ AND (time IS $THigh$ OR $TVeryHigh)]$ THEN $changeParameters$ of Meta-heuristic.

where:

- d_n is the difference between the benefit obtained by the meta-heuristic n and the one that is being studied divided by the best.
- the weights were obtained during the knowledge extraction process. The weights for each metaheuristic are a function of the error obtained by each one of them. This error is classified with the fuzzy sets: $zero$, $verysmall$, $acceptable$ and bad. After inferring the error obtained by every metaheuristic, each weight is calculated as the frequency of instances solved by each metaheuristic with an error classified as acceptable or lesser with respect to the instances solved by all the metaheuristics with this error. This frequency is biased according to the error, being $zero$ the best and $acceptable$ the worst.
- $Enough$ is a fuzzy set with trapezoidal membership function defined as follows:

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

  where a, b, c, d are 0.005, 0.01, 1, 1 respectively. This membership function, as the rest of membership functions described, was determined using trial and error.
- $High$ is a fuzzy set with trapezoidal membership function where a, b, c, d are 0.05, 0.1, 1, 1, respectively.
- $THigh$ is a fuzzy set with trapezoidal membership function where a, b, c, d are 0.4, 0.5, 0.7, 0.8, respectively.
- $TVeryHigh$ is a fuzzy set with trapezoidal membership function where a, b, c, d are 0.7, 0.8, 1, 1, respectively.
- $ChangeParameters$ is a function that change the parameters of a meta-heuristic for a new set of parameters which showed good performance during the process of knowledge extraction. In order to do that, for each kind of instance, we obtain an order over the different parameters based on the error that obtain and assign parameters according to this order.
- A rule is considered to be fired if its activation is bigger than an $\alpha-cut$, that can be configured.

The first rule tries to indicate how the position in the search space of the meta-heuristic with the worst behaviour can be changed for a position near to another meta-heuristic with a better behaviour. In order to change the solution of the worst meta-heuristic, we can take into account the following situations:

- The meta-heuristic that receives the solution is based on trajectories. The best solution obtained among the meta-heuristics that have fired the rule is sent to it.
- The meta-heuristic that receives the solution is based on populations. There are different options:
  - ◇ The meta-heuristic that sends the solution is based on trajectories. It has to send a set of solutions consisting of the replication of its best solution.
  - ◇ The meta-heuristic that sends the solution is based on populations. It has to send a set of solutions consisting of the best members of its population.
  - ◇ Different meta-heuristics has to send solutions. They have to send a set of solutions where each meta-heuristic choose its solutions as said before and they are combined paying attention to their weights.

On the other hand, the second rule shows how the parameters of the different meta-heuristics have to be modified. That way if a meta-heuristic is obtaining solutions with a benefit smaller than the rest, and it has been a long time since its parameters have been changed, then we can change its parameters for a new set using new values obtained during Data Mining phase.

System Prototype for Knapsack Problem

In order to test this idea we applied the knowledge extraction process to obtain a prototype of the system for solving knapsack problem, [3]. The prototype is composed of three meta-heuristics, a genetic algorithm, a tabu search and a simulated annealing. In Data Preparation phase we solved 2000 instances using these algorithms with different parameter configurations. In Data Mining phase we used a fuzzy decision tree, FID 3.4 [8] to fill the template, obtaining different sets of parameters for each meta-heuristic and their weights, 0.45 for the Genetic Algorithm, 0.3 for the Simulated Annealing and 0.25 for the Tabu Search.

The prototype of the system was implemented synchronously, that is, every communication is carried out at a given moment, previously specified. At this moment each meta-heuristic 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 [12]. 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 Impact of Fuzzy Sets in the System

In this section we want to test the benefits of using fuzzy rules in our system. That way, we will test that as the system is partially controlled by fuzzy sets we can obtain, for each $\alpha - cut$, a different set of rules, and thus, better results depending on instance type. To test that we executed different experiments using a system with crisp rules and systems using different values for the $\alpha - cut$ that controls the firing of the rules.

3.1 Test Database

To carry out the tests we solved a database of instances composed of 20 instances where changed both size (number of objects) and the way they were generated. Regarding size we can find four different sizes: 500, 1000, 1500 and 2000 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.

These instances were different from the instances used in the process of knowledge extraction. That way we can validate the model of the coordinator using a test database different from the training database.

3.2 Methodology

In order to compare the systems we decided that each one had to be executed during 120 seconds stopping each 250 milliseconds to execute the coordination. Each instance was solved 10 times, and the results show the averages. All tests were executed on an Intel core2 Quad 1.66Ghz with 2GB of Memory.

3.3 Results

The results of the tests can be seen on table 1, that shows the average error ratio, after 10 executions, obtained in the resolution of the different types of instances by each system, being highlighted the system which obtained the best performance. 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.

Table 1. Tests results

	Crisp	α =0.5	α =0.6	α = 0.7	α = 0.75	α = 0.8	α = 0.9
unc span	0,463	0,404	**0,33**	0,362	0,431	0,361	0,514
wea span	2,647	2,611	2,595	**2,556**	2,606	2,582	2,669
str span	0,913	0,917	0,913	0,920	**0,899**	0,923	0,919
pceil	0,114	**0,11**	0,115	0,116	0,116	0,116	0,114
circle	7,821	7,738	7,966	7,834	7,893	**7,724**	7,803
500	2,238	**2,102**	2,286	2,164	2,144	2,192	2,14
1000	2,745	2,748	2,816	2,766	2,833	**2,711**	2,850
1500	**3,205**	3,259	3,244	3,294	3,25	3,262	3,218
2000	3,771	3,677	3,572	3,564	3,719	**3,541**	3,811

We can check the better performance of the fuzzy systems, as only in one case the crisp system outperforms them. That way, for every type of instance the fuzzy systems obtained better results than the crisp one, and only for instances of size 1500 the crisp system obtained the best performance. With regard to the different $\alpha-cut$ values none can be said to be the best. However, by changing it we can obtain different behaviors and thus, the system becomes more tolerant and flexible when we try to solve different instances. Thus, we have found an interesting parameter for the system configuration. Because if we change $\alpha - cut$ we can obtain a better behaviour depending on the type of the instance being solved.

4 Conclusions and Work to Develop

During the development of this paper we have shown the construction of a hybrid, centralized, cooperative system and how the use of fuzzy technologies can improve its performance due to the elasticity that provide us.

The system is based on a multi-agent system, where each meta-heuristic is an agent that has to solve the problem while cooperates with the rest. In order to accomplish this coordination we have used a coordinating agent which controls and modifies the behaviour of the agents during their execution. To add intelligence to the coordinator we have applied a knowledge extraction process obtaining a set of fuzzy rules. With these rules the coordinator can:

- Change the position in the search space of a meta-heuristic which is obtaining poor results for another position near to the position of a meta-heuristic with a better performance.
- Change the parameters of a meta-heuristic intelligently if it persists in having bad results.

With the model of the coordinator we built a synchronous prototype of the system, that was used to solve a set of 20 instances, using crisp rules and

different values for the $\alpha - cut$ that controls the firing of the fuzzy rules. After this comparison we could observe how the fuzzy engine improved the performance of the system and became an important parameter. This parameter should be studied and configured for each type of instance.

It is important to outline that if we increment the number of metaheuristics cooperating the knowledge extraction process will take more time, but the performance of the system will not decrease, because each metaheuristic is independent from the others. Only the coordinator will be a bit slower, however, its execution time compared with the execution time of the meta-heuristics is negligible.

Regarding future work we have outlined that the $\alpha - cut$ is an important parameter for the design of the coordinator of our system because it can change its behaviour. For that reason we propose to obtain a decision tree that, added to the coordinator, will decide which $\alpha - cut$ has to be used with each instance. But the $\alpha - cut$ is not the only parameter from the template that needs tuning, for that reason we want to obtain models (based on decision trees) that will choose the parameter values that have to be used for each instance.

On the other hand, we can outline that the system has been applied to a very simple problem, knapsack problem, which is considered one of the "easiest" NP problems. For that reason we expect to apply this process to more complex problem, as the p-median, the p-hub median or the protein structure comparison.

It has also been noticed that the cost of the knowledge extraction process can be too large. To cope with this problem we propose two approaches:

- To use Active Learning to reduce the time expended in Data Preparation phase, as with this technique we can reduce the instances that need to be solved to generate the database.
- To use a system based on Online Learning. That way we will have an initial system with a "bad" performance that will be improving as it solves more instances.

Acknowledgements

The authors thank the "Ministerio de Educación y Ciencia" of Spain and the "Fondo Europeo de Desarrollo Regional" (FEDER) for the support given to this work under the project TIN2005-08404-C04-02. And the "Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia", which give support under the "Programa Séneca".

References

1. Cadenas J.M, Garrido M.C, Hernández L.D, Muñoz E (2006) Towards a definition of a Data Mining process based on Fuzzy Sets for Cooperative Metaheuris-

tic systems. International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU2006, 2828–2835, Paris
2. Cadenas J.M, Garrido M.C, Liern V, Muñoz E, Serrano E (2007) Un prototipo del coordinador de un Sistema Metaheurstico Cooperativo para el Problema de la Mochila. V congreso espaol sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados, MAEB07, 811–818, Tenerife, Spain
3. Cadenas J.M, Garrido M.C, Muñoz E (2007) A Cooperative System of Metaheuristics. 7th International Conference on Hybrid Intelligent Systems, HIS 2007, Kaiserslautern, Germany
4. Cohoon J, Martin W, Richards D (1991) A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes. In: Richar K. Velw, Lashon B. Booker (eds) Fourth International Conference on Genetic Algorithms, San Mateo. CA: Morgan Kaufmann Publishers
5. Crainic T.G., Toulouse, M.(2003) Parallel Strategies for Metaheuristics, Handbook of Metaheuristics, Kluwer Academic Publisher
6. Crainic T.G, Gendreau M, Hansen P, Mladenovic N (2004) Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10:293–314
7. Guo H (2003) A Bayesian Approach for Automatic Algorithm Selection. IJCAI03 Workshop on AI and Autonomic Computing, 1–5, Mexico
8. Janikow C.Z (1998) Fuzzy decision trees: issues and methods. IEEE Transaction System, Man, and Cybernetics, Part B. 28(1):1–14
9. Kuncheva, L.I. (2003) 'Fuzzy' vs 'Non-fuzzy' in combining classifiers designed by boosting. IEEE Transactions on Fuzzy Systems 11(6):729-741
10. Le Bouthillier A., Crainic T.G (2003) A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Computers and Operations Research 32(7):1685–1708
11. Lee k., Lee s. (1992) Efficient parallelization of simulated annealing using multiple markov chains: an application to graph partitioning. International Conference on Parallel Processing, 177-180, Michigan, USA
12. 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, Leicester, England
13. 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
14. Rice J.R (1976) The algorithm selection problem. Advances in Computers 15:65–118
15. University of Waikato. Weka, Data Mining with Open Source Machine Learning Software in Java. URL: http://www.cs.waikato.ac.nz/ml/weka/