Using Knowledge Discovery in cooperative strategies: two case studies

A.D. Masegosa, E. Muñoz, D.Pelta, and J.M. Cadenas

Abstract

1 Introduction

Although some algorithms have a good performance in a specific problem, there is hardly an algorithm which behaves better than others in a wide set of instances of such problem. This fact corresponds with the No Free Lunch Theorem [20]. In this way, it is very complicated to determine what the best method for a given instance is, specially if there are big differences in performance from one algorithm to another. This is known as the "Algorithm Selection problem" [19], defined by Rice in 1976.

This problem has been treated in various areas. One of them is Machine Learning [11, 5, 10]. This kind of techniques have been utilised to estimate the execution time required by an algorithm to solve a determined type of instances, so that through this information, we can choose the best expected method when we face a new instance. Another technique used to deal with this problem can be framed in the "Algorithm Portfolio" paradigm. Such paradigm consists of a set of algorithms which are executed until the fastest one solves the problem. An example of this type of strategies can be found in [15]. Within this paradigm, we can found the cooperative one which consist on a set of heuristic searches which exchange information among them. This collaboration leads to a dramatically improve in the robustness and the quality of the solutions obtained respect to the independent version of the strategy [2, 6]. This concept of cooperation is successfully used, explicit or implicitly, in other types of

{A.D. Masegosa, D. Pelta}
Dept. of Computer Science and Artificial Intelligence
University of Granada, Granada, Spain. e-mail: {admase, dpelta}@decsai.ugr.es

{E. Mu noz, J.M. Cadenas}
Dept. Ingeniera de la Informacin y las Comunicaciones
University of Murcia, Murcia, Spain. e-mail: enriquemuba@dif.um.es, jcadenas@um.es

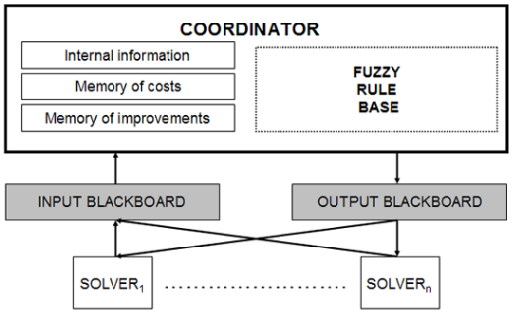


Fig. 1 Scheme of the cooperative strategy

metaheuristics as multi-agent systems (ACO's [8], PSO's[12]), memetic algorithms [13] and hyperheuristics [3].

In this paper we are going to treat with both areas, cooperative strategies and Machine Learning. Concretely, we will discuss to what extent and in what contexts the use of Knowledge Discovery techniques can improve the performance of the cooperative strategies. For this purpose, a centralised cooperative strategy based on simple elements of Soft Computing, previously presented in [7, 18, 17, 4], will be utilised as a base case. From this base case, we will analyse the improvement produced by the use of a new rule and a novel off-line parameter adjustment method, both of them obtained using Data Mining. For such analysis two different case studies will be considered. Both cases differ in terms of implementation and test bed used (Uncapacitated Single Allocation p-Hub Median Problem (USApHMP) and the Knapsack problem). We have chosen these two problems for the following reasons: the USApHMP is a NP-hard problem where only small datasets of solved instances can be found, and for that reason we have little information in order to perform a KD. On the other hand, Knapsack Problem is one of the "easiest" NP-hard problems, in which simple resolution algorithms obtain good results, and where we can find big datasets of solved instances. These test beds are two extreme situations in which we want to check the improvements obtained by the KD.

This work is structured as follows. Firstly, we will describe the two centralised cooperative strategy used as base case. In Section 2, the new two components obtained by Knowledge Discovery will be shown. Section 4 is devoted to explain the obtaining process of such components. After that, we will relate the experimentation done and the results obtained. To finish, in Section **??**, the conclusions of this work, will be discussed.

2 A centralised cooperative search strategy

In this cooperative strategy, whose general scheme is presented in Figure 1, there is a set of solvers, where each solver implements the same or different resolution strategy for the problem at hand. The coordinator processes the information received

from the solvers and, making use of a fuzzy rule base, produces subsequent adjustments of their behaviour by sending "orders". To achieve this exchange of data, a blackboard architecture [9] is used.

A important part of this strategy is the information flow. This is divided in the following three steps: 1) performance information is sent to the coordinator from the solvers, 2) this information is processed and stored by the coordinator and 3) coordinator sends orders to the solvers.

In the data flow from solvers to the coordinator, each report contains the next items:

- Solver identification
- A time stamp t
- The current solution of the solver at that time s^t
- The best solution reached until that time by this solver s_{best}

The coordinator keeps the last two reports sent by each solver, so in the next step (information processing), the improvement rate is calculated as $\Delta_f = \frac{f(s^t)-f(s^{t-1})}{time^t-time^{t-1}}$, where $time^t - time^{t-1}$ represents the elapsed time between two consecutive reports and f is the objective function. The values Δ_f and $f(s^t)$ are then stored in two fixed length ordered "memories", one for improvements and another for costs.

Over those memories, the next fuzzy rule is constructed. This rule is used by the coordinator to determine if a solver is working fine or not. It was designed based on expert knowledge following the principle: *If a solver is working well, keep it*; but *if a solver seems to be trapped, do something to alter its behaviour*. From now on, this rule are going to be called EK rule and its definition is the next one:

IF the quality of the current solution reported by $solver_i$ is low AND the improvement rate of $solver_i$ is low **THEN** send C_{best} to $solver_i$

The label *low* is defined as a fuzzy set with the following membership function:

$$\mu(x,\alpha,\beta) = \begin{cases} 0, & \text{if } x > \beta \\ \frac{\beta-x}{\beta-\alpha}, & \text{if } \alpha \leq x \leq \beta \\ 1, & \text{if } x < \alpha \end{cases}$$

where x is the relative position (resembling the notion of percentile rank) of a value (an improvement rate or a cost) in the samples stored in memory of improvements or memory of costs, respectively, and the other parameters are fixed to $\alpha = 80$ and $\beta = 100$ for the memory of costs, and $\alpha = 0$ and $\beta = 20$ for the memory of improvements. C_{best} denotes the best solution ever recorded by the coordinator. In short, what the rule says is that if the values reported by a solver are among the worst in the memories, then such a solver should be changed in some way. By means of the C_{best} sending, as the strategy progresses, the solvers will therefore concentrate around the most promising regions of the search space, which will be sampled using different schemes (the ones defined by the solver threads themselves). This increases the chances of finding better and better solutions.

Depending on the nature of the solvers (based on trajectories or in populations),the sending of C_{best} is achieve in a different way. For trajectory based methods, this solution is altered by a mutation operator before be sent to the solvers. This modified solution is consider as a start point of the search. Such alteration tries to avoid relocating all solvers in the same point of the search space. However, for population based methods, a proportion of the worst individuals of the receiver is substituted by a set of solutions obtained altering C_{best} using the same mutation operator.

In the following section we are going to describe the two components that will be incorporated in this strategy to try to improve its performance: a new rule, named KD rule, and a new off-line adjustment parameter method.

3 New components obtained by Knowledge Discovery

The first described element is the KD rule. We should say that this rule is not completely designed using Knowledge Discovery techniques, but it is a template that is completed by means of a Data Mining process that will be described in 4. 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 complete operative system, without the need of applying the knowledge extraction process again.

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

- IF [$solver_i$ IS $theWorst$] AND [($wm_1 * d_1$ OR ... OR $wm_n * d_n$) IS $enough$] THEN change the current solution of $solver_i$.
- IF [($wm_1 * d_1$ OR ... OR $wm_n * d_n$) IS $High$ AND ($time$ IS $THigh$ OR $TVeryHigh$)] THEN $changeParameterValues$ of $solver_i$.

where:

- n is the number of solvers.
- $solver_i$ is the solver being evaluated by the rule.
- $theWorst$ evaluates if the solver being studied now is having the worst performance according to any previously defined measure.
- $d_i = (perf_i - perf_{MH})/maximum(perf_i, perf_{MH})$, 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 metaheuristic i (importance of metaheuristic i for solving the current instance).
- $Enough$, $High$, $THigh$ and $TVeryHigh$ are fuzzy sets 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):

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

– *ChangeParameterValues* is a function that changes the values of the parameters of a solver. To do this, a model is used which gives an order to the combinations of parameter values for the instance being solved. In this way the first set of parameter values used by the system is the first of this order and when a change of parameter values is needed the next set, in order, is used and so on.

The first of these rules changes the position in the search space of a thread which is showing a bad performance for a position close to the position of another metaheuristic with a better behavior. The second rules shows how the values of the parameters of the different metaheuristics have to be modified. The values of the wm_i's is the part completed by Knowledge Discovery techniques.

The off-line adjustment parameter method uses a classification tree also obtained by this type of techniques to determine the most suitable parameters of each heuristic that compose the strategy, depending on the instance characteristics.

4 Data Mining process used for the new components

In this section, we will describe briefly the methodology used, previously proposed on [4]. Such methodology uses some fuzzy decision trees in order to obtain the wm_i $\forall i$ that define the rules and the parameters values of each solvers. These trees receive a description of the instance that has to be solved and output the wm_i and the order among the combination of parameter values for each meta-heuristic. That way we have a tree that assigns the weights to each meta-heuristic and as many trees as meta-heuristics to obtain the order among the combination of parameter values to be used, and those that replace them. The idea of how the trees give the configuration of the set of fuzzy rules can be seen on fig. 2.

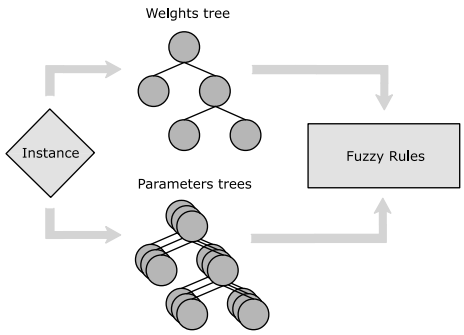


Fig. 2 Obtaining of weights and parameters values for meta-heuristics.

	Case study 1	**Case study 2**
Communication mode	asynchronous	synchronous
Stop condition and communication frequency	evaluation number	Time
Implemented solvers	VND, Tabu, SA	Tabu, SA, GA
Test problem	USApHMP	knapsack
Amount of instances	34	2000
Amount of instances for training	34	500
Amount of instances/size and type	1	25
Amount of instances for test	34	20

Table 1 Main differences between the two implementations and the two problems

To obtain these tree models we use a knowledge extraction process which is divided in two main phases: Data Preparation and Data Mining. In Data Preparation phase we seek to obtain a database which holds 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 obtain the trees previously described.

5 Case studies details

This section is devoted to describe the two case studies used in the experimentation. When we say case study, we refer to a pair implementation-problem. In this way, we dispose of two distinct pairs that differs in both implementation and problem. The next two subsections will show the details of each case study. To remark the principal existing differences between the two case studies, these are displayed in Table 1 as a summary.

5.1 Case study 1 description

When implementing multi-threaded cooperative strategies, one can resort to parallel schemes if time is important, or one can simulate the parallelism in a one-processor computer. This is the strategy taken here and the procedure is extremely simple. We construct an array of solvers and we run them using a round-robin schema. This implementation uses an asynchronous communication mode that is simulated in this way: solvers are executed during a certain number of evaluations of the objective function. This amount of evaluations is a random number in the interval $[100, 150]$ and after this period of time, information exchanges are performed. After the execution of all solvers, the coordinator is launched to do its corresponding functions.

These steps are repeated until the stop condition, given in terms of objective function evaluations, is fulfilled.

Regarding the fuzzy rule base, firstly, we should mention that the size of the memory of costs and improvements was set to be double the number of solvers. The modification applied to the best solution sent to the threads consists on a change of assignment of non-hub nodes and a change of localisation of hub nodes. Three different heuristic searches were chosen as solvers: Tabu Search, Simulated Annealing (SA) and Variable Neighborhood Descent search(VND). The description of this methods is omitted due to space constraints.

The test bed used in this case is a hub location problem. The objective of this type of problems is composed of two steps: 1)**Hub location:** to determine which nodes should be the hubs and the number of them, in order to distribute the flow across them, and 2) **Non-hub to hub allocation:** to assign the rest of the nodes to the hubs. Generally, these tasks are performed by minimising an objective function that describes the exchange flow and its cost. We will focus on a special case, the Uncapacitated Single Allocation p-Hub Median Problem (USApHMP), which is consider as a NP-hard problem. Its quadratic integer formulation was given by O'Kelly in [16].

The instances chosen for the experimentation were obtained from the resource ORLIB [1]. Concretely, we used the AP (Australian Post) data set derived from a study of the postal delivery system. The instances utilised can be divided in two groups, those with 50 or less nodes, and those with more than 50. For the first set, instances with 2, 3, 4 and 5 hubs were considered , while for the second one, the different numbers of hubs were 2, 3, 4, 5,10,15 and 20. The optimum for those instances with a number of nodes less than 50 was provided by the resource ORLIB, and for the other instances we considered the best solution found for one of the state-of-art algorithms for this problem, presented in [14]. In order to asses the quality of the solutions returned by the strategy, we consider an error as $error = 100 \times \frac{obtained\ value - optimum}{optimum}$. From the point of view of the Knowledge Discovery process, we should highlight that in this case we only have a total of 34 instances and there is just one instance per type and size.

5.2 Case study 2 description

In this case study the implementation strategy is the same that in the previous one. But in this case the communication mode is synchronous stopping each strategy after 100 ms, after the execution of all solvers, the coordinator is launched to do its corresponding functions. These steps are repeated until the stop condition, given in terms of time, is fulfilled.

Three different heuristic searches were chosen as solvers: Genetic Algorithm (GA), Tabu Search (TS) and Simulated Annealing (SA). The description of this methods is omitted due to space constraints.

The test bed used in this case is the well known knapsack problem. This problem 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.

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.

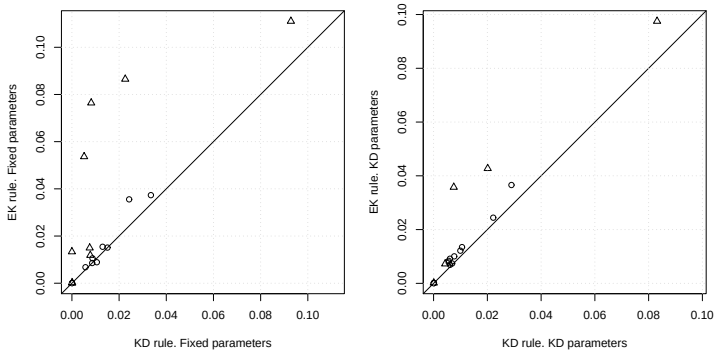
In order to asses the quality of the solutions returned by the strategy, we consider an error as $error = 100 \times \frac{obtained\ value - optimum}{optimum}$. From the point of view of the Knowledge Discovery process, we should highlight that in this case we had an instance generator and for that reason we were able to perform the training of the system using 500 instances, with 25 instances per type and size.

6 Experiments and results

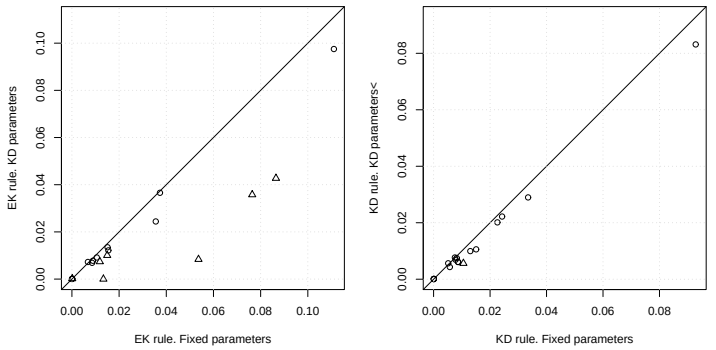
The experimentation done in this paper has as target to analyse the benefits contributed by the Data Mining process seen in Section 4. The two main aspects that will be studied are the followings:

- Profits obtained by the use of the KD rule instead of the EK rule.
- Improvement achieved by the off-line adjustment parameter method.

Both aspects will be test on the two case studies we saw in last section, so the experimentation are going to be structured in four different result analysis, that will be discussed in the rest of the section. Before starting with this part, we will state some details about the experimentation and the terminology used in this section. Firstly, the number of runs per instance is 30 for p-hub instances and 10 for knapsack instances, and secondly, when we refer to statistically significant differences or improvements, these have been calculated using the Mann's Whitney non-parametric test with $\alpha < 0,05$.



a) Fixed parameters b) off-line KD method



a) EK rule b) KD rule

6.1 Profits obtained by the use of the KD rule versus the EK rule

In fig. 6.1 two graphics are shown in which we want to compare the performance of the KD rule with respect to the EK rule. In both graphics we can see how in general the use of the KD rule obtains a better performance, however in some cases the differences are not significant. We have to notice that although Knapsack problem is easy to solve, the KD has obtained an improvement in the solutions.

6.2 Improvement achieved by the off-line adjustment parameter method

In fig. 6.2 we compare the use of the off-line adjustment of parameter versus the fixed adjustment. In this case we can observe how the use of the off-line adjust-

ment obtains also an improvement of the results, despite the fact of being Knapsack problem an easy one, however in some cases the differences are not significant.

Acknowledgements A.D. Masegosa is supported by the scholarship program FPI from the Spanish Ministry of Science and Technology. This work has been partially funded by the project TIN-2005-08404-C04-01 from the Spanish Ministry of Science and Technology and TIC-00129-PE from the Andalusian Government. 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 to E. Muñoz under the "Programa Séneca".

References

1. J. Beasley. Obtaining test problems via internet. *Journal of Global Optimization*, 8(4):429–433, 1996.
2. A. L. Bouthillier and T. G. Crainic. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Comput. Oper. Res.*, 32(7):1685–1708, 2005.
3. E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. *Handbook of meta-heuristics*, chapter Hyper-heuristics: an emerging direction in modern search technology, pages 457–474. Kluwer Academic Publishers, 2003.
4. J. Cadenas, M. Garrido, L. Hernández, and E. M. noz. Towards a definition of a data mining process based on fuzzy sets for cooperative metaheuristic systems. In *Proceedings of IPMU2006*, pages 2828–2835, 2006.
5. T. Carchrae and J. C. Beck. Applying machine learning to low-knowledge control of optimization algorithms. *Computational Intelligence*, 21(4):372–387, 2005.
6. T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenović. Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10(3):293–314, 2004.
7. C. Cruz and D. Pelta. Soft computing and cooperative strategies for optimization. *Applied Soft Computing Journal*, 2007. doi:10.1016/j.asoc.2007.12.007. In press.
8. M. Dorigo and T. Stützle. *Ant Colony Optimization*. Bradford Book, 2004.
9. J. Ferber. *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1999.
10. H. Guo and W. H. Hsu. A machine learning approach to algorithm selection for np-hard optimization problems: a case study on the mpe problem. *Annals of Operations Research*, 156(1):61–82, 2007.
11. E. Houstis, A. Catlin, J. R. Rice, V. Verykios, N. Ramakrishnan, and C. Houstis. Pythia-ii: a knowledge/database system for managing performance data and recommending scientific software. *ACM Transactions on Mathematical Software*, 26(2):227–253, 2000.
12. J. Kennedy and R. C. Eberhart. *Swarm intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001.
13. N. Krasnogor and D. A. Pelta. *Fuzzy Sets based Heuristics for Optimization*, volume 126 of *Studies in Fuzziness and Soft Computing*, chapter Fuzzy Memes in Multimeme Algorithms: a Fuzzy-Evolutionary Hybrid, pages 49–66. Springer-Verlag Berling Heidelberg New York, 2002.
14. J. Kratica, Z. Stanimirović, and V. F. Dušcan Tovšić. Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem. *European Journal of Operational Research*, 182(1):15–28, 2007.
15. K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham. Boosting as a metaphor for algorithm design. *Lecture Notes in Computer Science*, 2833:899–903, 2003.

16. M. O'Kelly and E. Morton. A quadratic integer program for the location of interacting hub facilities. *European Journal of Operational Research*, 32(3):393–404, 1987.
17. D. Pelta, C. Cruz, A. Sancho-Royo, and J. Verdegay. *Fuzzy Applications in Industrial Engineering*, volume 201 of *Studies in Fuzziness and Soft Computing*, chapter Fuzzy Sets based Cooperative Heuristics for Solving Optimization Problems. Springer, 2006.
18. D. Pelta, A. Sancho-Royo, C. Cruz, and J. L. Verdegay. Using memory and fuzzy rules in a cooperative multi-thread strategy for optimization. *Information Sciences*, 176(13):1849–1868, 2006.
19. J. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976.
20. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.