Using Knowledge Discovery in Cooperative Strategies: Two Case Studies

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

Abstract. In this work we discuss to what extent and in what contexts the use of knowledge discovery techniques can improve the performance of cooperative strategies for optimization. The study is approached over two different cases study that differs in terms of the definition of the initial cooperative strategy, the problem chosen as test bed (Uncapacitated Single Allocation p Hub Median and knapsack problems) and the number of instances available for applying data mining. The results obtained show that this techniques can lead to an improvement of the cooperatives strategies as long as the application context fulfils certain characteristics.

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 [21]. 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. Formally, this is known as the "Algorithm Selection problem" [20], and was defined by Rice in 1976.

This problem has been treated in various areas. One of them is Machine Learning [5, 11, 12]. These kind of techniques have been used to estimate the execution time required by an algorithm to solve a determined type of instances, so that through this

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. Muñoz · J.M. Cadenas
Dept. Ingeniería de la Información y las Comunicaciones
University of Murcia, Murcia, Spain
e-mail: enriquemuba@dif.um.es, jcadenas@um.es

information, we can choose the best expected method when we face a new instance. Another technique is associated with the "Algorithm Portfolio" paradigm, where, instead of selecting a single algorithm, a set of methods are executed in parallel until the fastest one solves the problem. An example of this type of strategies can be found in [17]. When the algorithms are allowed to exchange information among them, then cooperative search strategies arise, and this collaboration leads to a dramatically improve in the robustness and the quality of the solutions obtained with respect to the independent version of the strategy [2, 6]. This concept of cooperation is successfully used, explicit or implicitly, in other types of metaheuristics as multi-agent systems (ACO‘s [8], PSO‘s[14]), memetic algorithms [15] 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 cooperative strategies. For this purpose, a centralised cooperative strategy based on simple elements of Soft Computing, previously presented in [4, 7, 19], will be consider as the baseline case. From this starting point, we will analyse the improvement produced by the use of new control rules and two alternatives for setting the initial parameters of the methods composing the cooperative strategy. These features are obtained using data mining. The study will be conducted on two different scenarios that differ in terms of the baseline 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 the training phase in the knowledge discovery process. 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 for training the system. 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 centralised cooperative strategy used as base case. In Section 3, the new control rule and the two types of initial parameter tune will be shown. Section 4 is devoted to state the two case studies used to test the cooperative method. After that, we will relate the experimentation done and the results obtained. To finish, in Section 6, the conclusions of this work, will be discussed.

2 A Centralized Cooperative Search Strategy

The cooperative strategy described in [7, 19], consists on a set of solvers/threads, each one implementing the same or a different resolution strategy for the problem at hand. These threads are controlled by a coordinator which processes the information received from them and, making use of a fuzzy rule base, produces subsequent adjustments of solver behaviours by sending "orders". The information exchange process is done through a blackboard architecture [9].

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

Each report in the first step contains:

- 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 stores the last two reports from each solver, so in the information processing step, the improvement rate is calculated as $\Delta_f = \frac{f(s^t)-f(s^{t'})}{t-t'}$, where $t-t'$ represents the elapsed time between two consecutive reports, $s^{t'}$ is the current solution sent by the solver in the last report 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.

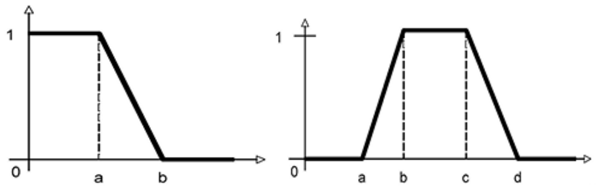
Over those memories, a fuzzy control rule is constructed. This rule allows the coordinator to determine if a solver is working fine or not. It was designed based on expert knowledge following the principle: *If a solver is working well, keep it*; but *if a solver seems to be trapped, do something to alter its behaviour*. From now on, this rule is called *EK* 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 whose membership function $\mu(x)$ is shown in Figure 1 (a). The variable x will correspond with 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 $a = 80$ and $b = 100$ for the memory of costs, and $a = 0$ and $b = 20$ 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 sending C_{best}, it is expected that the solvers will 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 (trajectory-based or population-based), the solution C_{best} is sent in a different way. For trajectory based methods, a new solution C'_{best} is obtained from C_{best} using a mutation operator. When the solver receives C'_{best}, then it will restart the search from that new point. Such modification tries to avoid relocating all of the solvers in the same point of the search space.



a) Definition of *low* b) Definition of enough, High, THigh and TVeryHigh

Fig. 1 Definition of low, enough, High, THigh and TVeryHigh

However, for population based methods, a proportion of the worst individuals of the receiver is substituted by a set of mutated solutions obtained from C_{best} using the same operator as before.

3 Knowledge Discovery for Rule Design and Parameters Setup

Any of the components defining the basic cooperative strategy could be changed. In this work, we will consider new definitions for two relevant components: 1) a new set of control rules and 2) a mechanism to setup the initial parameters of the threads. Both features will be obtained using knowledge discovery techniques that are fully described in [4]. The basic ideas of the process is briefly presented here.

The first step is the data generation process where we have:

- $\{m_0,\ldots,m_k\}$, a set k metaheuristics
- $\{c_{i,0},\ldots,c_{i,d}\}$, a set of possible parameter combinations for m_i
- $\{p_0,\ldots,p_l\}$ the set of training instances

Then, every m_i is run over each p_t with every possible combination of parameters c_{ij} in order to obtain a performance information database. The second step in the knowledge discovery process is to extract several decision trees. Then, when the cooperative system is presented with a new instance to solve, the trees are traversed and certain weights for the control rules are returned (from the “Weights Tree”), and a list of “good parameter configurations” is constructed (from the “Parameters Tree”). From this list, the system will setup the parameters of the threads. See Figure 2 for a schematic description.

3.1 New Set of Control Rules

The new set of control rules has two parameterized rules: the first one allows to change the position in the search space of a thread (because it may show a bad performance) making it closer to the one of another metaheuristic with a better behavior; the second rule allows to dynamically change the parameters governing the behaviour of a thread.

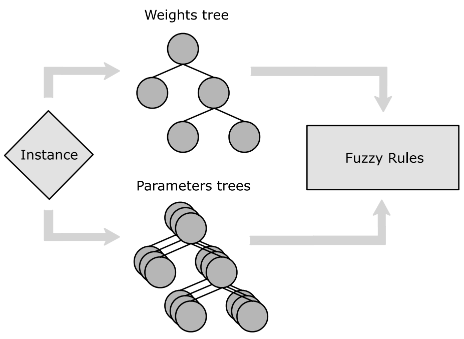


Fig. 2 Given a new instance to solve, new control rules weights and solvers' initial parameters are calculated

The parameterized rules (one for each thread or metaheuristic) are as follows:

- 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.
- $w_{m_i} \in [0,1]$ where $\sum_{i=1}^{n} w_{m_i} = 1$ and w_{m_i} represents the weight of solver i (importance of metaheuristic i for solving the current instance). These weights are calculated from the "Weights Tree" obtained from the data mining process.
- $Enough$, $High$, $THigh$ and $TVeryHigh$ are fuzzy sets with trapezoidal membership functions with support contained in $[0,1]$ defined by a cuadruplet (a,b,c,d). Its representation is shown in Figure 1b).
- $ChangeParameterValues$ is a function that changes the values of the parameters of a solver. As stated before, a list of "good parameter configurations" was obtained from the "Parameters Tree". So, when the rule is triggered, the next configuration from the list is selected.

3.2 Parameters Adjustment

The initial parameters of the solvers are calculated from the "Parameters Tree". In fact, there exist to different operational modes. In the first one, the parameters are calculated as a function of the type of the instance, while in the second one, the parameters are independent from the instance being solved. In this last case, the best configuration of parameters is the one that allowed to obtain, on average,

Table 1 Main features of the two case studies proposed

	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
Number of instances	34	2000
Number of instances for training	34	500
Number of instances per size and type	1	25
Number of instances for test	34	20

the best results over the set of training instances. Both strategies are considered in this work.

4 Case Studies Details

This section is devoted to describe the two case studies designed to assess to what extent and in what contexts the use of knowledge discovery techniques can improve the performance of cooperative strategies. The scenarios differ in the type of basic cooperative strategy used, in the communication model, test problem and information available for the data mining stage. The next two subsections fully describe the two scenarios, while their main features are displayed in Table 1.

4.1 Case Study 1

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 a synchronous communication mode that is simulated in this way: solvers are executed during 100 ms each one and after this period of time, information exchanges are performed. These steps are repeated until the stopping condition, given in terms of running time, is fulfilled.

Regarding the fuzzy rule (EK rule), the size of the memory of costs and improvements was set to be double the number of solvers. Three different heuristic searches were chosen as solvers: Genetic Algorithm (GA), Tabu Search (TS) and Simulated Annealing (SA). Their implementation follows the basic guidelines described in [10] and no specific tailoring of operators to problem was done. The description of these methods is omitted due to space constraints.

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

From the point of view of the Knowledge Discovery process, and due to the availability of an instance generator, the number of instances available for training the system was 500, with 25 instances per size and type considered. We used four different sizes: 500, 1000, 1500 and 2000 objects and five types of instances, given by Dr D. Pisinger in[13], were taken into account:

- 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.

To carry out the tests we solved a database of instances composed of 20 instances (one per type and size). In order to asses the quality of the solutions returned by the strategy, we consider an error as $error = 100 \times \frac{obtained\ value - optimum}{optimum}$.

4.2 Case Study 2 Description

In this case study, the implementation is broadly the same with some slight variations. Firstly, here the communication mode is asynchronous and is not determined by CPU time but by objective function evaluations. Concretely, the solvers are run during a random number of evaluations that varies from 100 to 150. The process is repeated until a maximum number of objective function evaluations has been done.

Other important difference with respect to the former one are the heuristic implemented by the solvers, since now all of them are trajectory based. The three different heuristic searches chosen were: Tabu Search, Simulated Annealing (SA) and Variable Neighborhood Descent search (VND). As before, their implementation follows the basic guidelines described in [10] and no specific tailoring of operators to problem was done.

The test bed is a hub location problem. The aim on this type of problems is composed of two steps: 1) **Hub location:** to determine which and how many nodes should be the hubs, 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 minimizing an objective function that describes the exchange flow and its cost. We will focus on a particular 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 [18].

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 a postal delivery system. The data set contains a first group of instances with 10,20,25,40 and 50 nodes (having 2,3,4 hubs), and a second group where the instances have 100 and 200 nodes with 2, 3, 4, 5,10,15 and 20 hubs. 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 [16]. The quality of the solutions is measured as in the previous case study. To finish this section, we should remark other significant distinction with respect to the case 1, since this one only have available a total of 34 instances and there is just one instance per type and size.

5 Experiments and Results

The experimentation done in this paper has as target to analyse the benefits contributed by the Knowledge Discovery process seen in Section 3. For this purpose, the baseline for comparison is the strategy seen in Section 2, where there is just one control rule (EK Rule) and the initial parameters for all the threads are those that gave the best results when averaged over all the training instances. In other words, they are independent from the instance being solved. The following combinations will be tested:

- KD rule is used instead of the EK rule.
- The parameters are calculated as a function of the instance being solved.

In this way, from the base case we can obtain the next strategies:

$$basic\ strategy \begin{cases} EK\ rule \begin{cases} parameters: instance\,independent(EK+IIP) \\ parameters: instance\,dependent(EK+IDP) \end{cases} \\ \\ KD\ rule \begin{cases} parameters: instance\,independent(KD+IIP) \\ parameters: instance\,dependent(KD+IDP) \end{cases} \end{cases}$$

Each one of the four cooperative strategies obtained is run over a set of test instances for every case study. We will first analyse the impact of the KD rule and then, that of the parameter's setting mode.

5.1 On the Impact of KD Rule versus EK Rule

Here, we compare the behaviour of the strategies EK+IIP vs. KD+IIP and EK+IDP vs. KD+IDP on each case study.

We will start the analysis with the first case study. Figure 3 shows two scatter plots where EK and KD rules are compared for both types of parameter's adjustment considered. In the scatter plots, each point represents a test instance and shows the relative deviation from the optimum for the two strategies compared. This is defined as $d = \frac{q-q^*}{q^*}$. Each point is an average over the total of runs. In this type of plots,

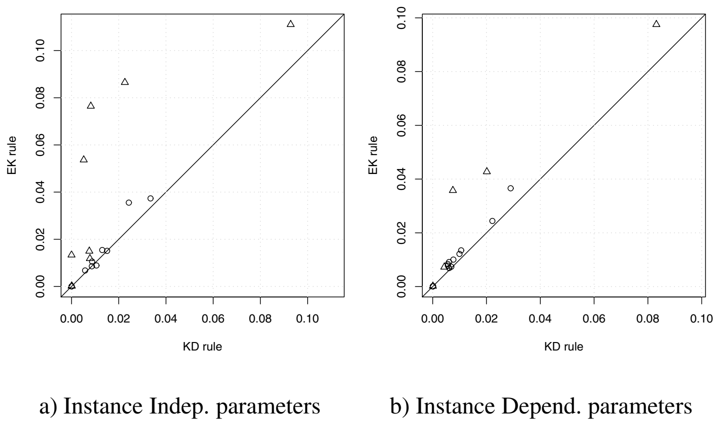


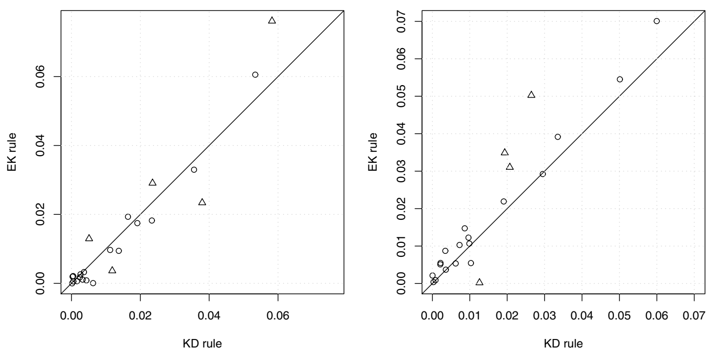
Fig. 3 Case Study 1: Comparison of the average relative deviation from optimum (smaller values are better). Triangles represent the instances on which the algorithms being compared perform differently at significance level 0.05 (Mann-Whitney U test(Wilcoxon's unpaired rank sum test))

when a point is below the diagonal line means that the strategy of the X axis has a worse average value than the strategy of the Y axis, and viceversa. When a point is represented by a triangle indicates that the difference is statistically significant (confidence level 0.05 by a Mann-Whitney U-test (Wilcoxon's unpaired rank sum test)) whereas in the opposite case, the point is showed as a circle.

Figure 3 (a) shows an important improvement when the KD rule is used with respect to EK. The cooperative strategy coupled with the KD rules always obtained equal or better average values (except in one instance). Moreover, the differences were statistically significant for 9 cases. When the parameters are set in terms of the type of the instance being solved, Figure 3 (b), the results are very similar. There is no point below the diagonal and the number of significant differences here is 5. In short, we can conclude that for this case study, the basic cooperative strategy can be enhanced with data mining techniques.

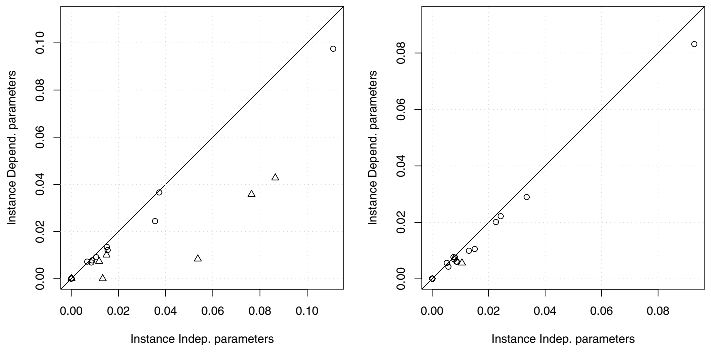
For the second case study, we are going to follow the same analysis structure. Figure 4 a) shows the performance of the EK rule vs the KD rule when both strategies use instance independent parameters. The differences in terms of results between the two rules are only statistically significant in 5 instances, three of which are positives for KD and the other two for EK. In the rest of the cases, the results are almost the same.

When the parameters are tuned accordingly with the type of instance, Figure 4b), there seems to be a slight improvement when using KD rules with respect to EK. Now, KD overcomes EK in most of the instances, being three cases statistically significant whereas this condition is only fulfilled in one occasion when such difference has the opposite sign. However, this result should be carefully analysed as the improve is not due to an enhancement of the KD rule, but a deterioration of EK, as we will see in the next subsection.



a) Instance Indep. parameters b) Instance Depend. parameters

Fig. 4 Case Study 2: comparison of the average relative deviation from optimum (smaller values are better). Triangles represent the instances on which the algorithms being compared perform differently at significance level 0.05 (Mann-Whitney U test (Wilcoxon's unpaired rank sum test))

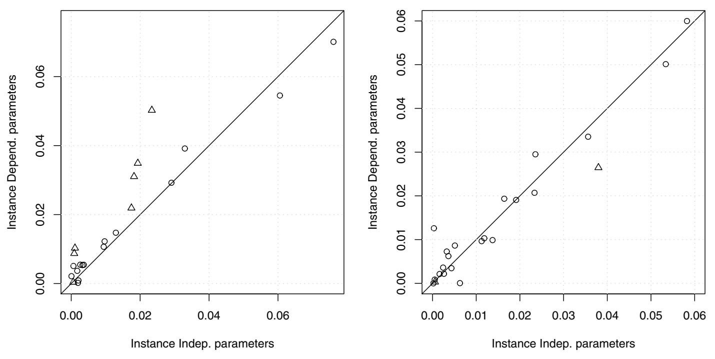


a) EK rule b) KD rules

Fig. 5 Case Study 1:Comparison of the average relative deviation from optimum (smaller values are better). Triangles represent the instances on which the algorithms being compared perform differently at significance level 0.05 (Mann-Whitney U test (Wilcoxon's unpaired rank sum test))

5.2 On the Impact of the Parameters Setup Method

This part of the result analysis is devoted to study to what extent the strategy improve its performance when the parameters of the heuristic are tuned as a function of the instance characteristics, so we will focus on EK+IIP vs. EK+IDP and KD+IIP vs. KD+IDP for both case studies.



a) EK rule b) KD rules

Fig. 6 Case Study 2:Comparison of the average relative deviation from optimum (smaller values are better). Triangles represent the instances on which the algorithms being compared perform differently at significance level 0.05 (Mann-Whitney U test(Wilcoxon's unpaired rank sum test))

In the first case study, the new parameter set up mechanism leads to a performance improvement that is very notorious for the EK rule, as we can see in Figure 5 a). The improvement achieved by the instance dependent parameter setting is significant in 7 instances and never drives the search to a deterioration. However, this enhancement is less appreciable for the KD rule. Viewing Figure 5 b), we can check that although the strategy always work better when the parameter are adjusted by this method, now the difference with respect to the other alternatives only statistically significant in one case.

For the second case study, we will start with the EK rule. Viewing the results shown by Figure 6a), we can observe the behaviour we pointed out before. The use of EK+IDP produce a high performance degradation of the basic strategy leading to worse results (the difference is statistically significant) in six instances.

When the control of the strategy is carried out by the KD rules, we can observe in Figure 6b) that the performance of KD+IIP and KD+IDP are almost the same.

6 Discussions

In this work we have seen how and in what contexts, Knowledge Discovery can be used to improve a centralised cooperative strategy. Concretely, the Knowledge Discovery has been incorporated in two different ways:

- By means of new parameterized control rules, where the parameters are determined using Data Mining
- Defining alternatives for setting up the parameters governing the behaviour of the metaheuristics: instance independent and instance dependent parameters that are provided, for each metaheuristic, by a decision tree.

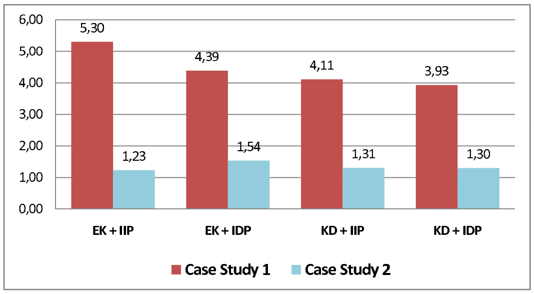


Fig. 7 Average error over all the instances of the corresponding case study, for every strategy evaluated

In order to analyse the suitability of the methodology, we proposed two case studies that differs in terms of the definition of the basic cooperative strategy (implemented heuristics, communication mode, ...), problem type, and amount of information available for doing knowledge extraction.

In the first case study, we observed that these new components led to cooperative strategies (EK+IDP, KD+IIP, KD+IDP) whose performance is better than the basic strategy (EK+IIP). This is clearer if we look at Figure 7 (case study 1) where the average error over all the test instances is shown for every strategy. This nice and clear behaviour is not present in the second case study.

In our opinion, the difference is related with amount of available information to "learn" in each case study. In other words, with the number of available instances to generate the performance information that then, should be mined to extract the weights and parameters that will govern the cooperative system. As we saw formerly, in the second case study we only had 34 instances for training with a unique sample per size and type, very low values to achieve a robust learning, specially if they are compare with such values in the first case study: 500 and 20 respectively.

Nevertheless, some conclusions can be obtained. First one is: if enough information is available to apply Knowledge Discovery techniques, then better cooperative strategies can be obtained. In second place, the benefit of using an instance dependent parameter setting needs to be further analysed because it depends on how well the instances in the training set could be characterized. If not enough information is available, then it will be safer not to use it. In the contrary, the use of KD rules when combined with an instance independent parameter setting leads to cooperative strategies that, at least, are as good as those using an expert designed rule for both case studies.

As future work, we plan to improve the learning process in order to reduce the amount of information needed to obtain meaningful knowledge. Another line of research consist on using online learning instead of the current offline data generation and processing method. In this way, the overhead of the learning process will be reduced and the future comparison against state of the art algorithms for specific problems could be fairly done.

Acknowledgements. A.D. Masegosa is supported by the scholarship program FPI from the Spanish Ministry of Science and Innovation. E. Muñoz is supported by "Fundación Séneca, Agencia de Ciencia y Tecnología de la Región de Murcia", under "Programa Séneca" action. This work has been partially funded by the projects TIN2008-01948 and TIN2008-06872-C04-03 from the Spanish Ministry of Science and Innovation and the "Fondo Europeo de Desarrollo Regional" (FEDER). Support from Andalusian Government through project P07-TIC-02970 is also acknowledged.

References

[1] Beasley, J.: Obtaining test problems via internet. Journal of Global Optimization 8(4), 429–433 (1996)

[2] Bouthillier, A.L., Crainic, T.G.: A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. Comput. Oper. Res. 32(7), 1685–1708 (2005)

[3] Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. In: Handbook of metaheuristics, pp. 457–474. Kluwer Academic Publishers, Dordrecht (2003)

[4] Cadenas, J., Garrido, M., Hernández, L., Muñoz, E.: Towards a definition of a data mining process based on fuzzy sets for cooperative metaheuristic systems. In: Proceedings of IPMU 2006, pp. 2828–2835 (2006)

[5] Carchrae, T., Beck, J.C.: Applying machine learning to low-knowledge control of optimization algorithms. Computational Intelligence 21(4), 372–387 (2005)

[6] Crainic, T.G., Gendreau, M., Hansen, P., Mladenović, N.: Cooperative parallel variable neighborhood search for the p-median. Journal of Heuristics 10(3), 293–314 (2004)

[7] Cruz, C., Pelta, D.: Soft computing and cooperative strategies for optimization. Applied Soft Computing Journal (2007) (In press) doi:10.1016/j.asoc.2007.12.007

[8] Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Book (2004)

[9] Ferber, J.: Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence. Addison-Wesley Longman Publishing Co., Inc, Boston (1999)

[10] Glover, F.W., Kochenberger, G.A. (eds.): Handbook of metaheuristics. Kluwer Academic Publishers, Dordrecht (2003)

[11] Guo, H., Hsu, W.H.: 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)

[12] Houstis, E., Catlin, A., Rice, J.R., Verykios, V., Ramakrishnan, N., Houstis, C.: Pythia-ii: a knowledge/database system for managing performance data and recommending scientific software. ACM Transactions on Mathematical Software 26(2), 227–253 (2000)

[13] Kellerer, H., Pferschy, U., Pisinger, D.: Knapsack Problems. Springer, Heidelberg (October 2004)

[14] Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)

[15] Krasnogor, N., Pelta, D.A.: Fuzzy Memes in Multimeme Algorithms: a Fuzzy-Evolutionary Hybrid. In: Fuzzy Sets based Heuristics for Optimization. Studies in Fuzziness and Soft Computing, vol. 126, pp. 49–66. Springer, Heidelberg (2002)

[16] Kratica, J., Stanimirović, Z., Dušcan Tovšić, V.F.: Two genetic algorithms for solving the uncapacitated single allocation p-hub median problem. European Journal of Operational Research 182(1), 15–28 (2007)

[17] Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., Shoham, Y.: Boosting as a metaphor for algorithm design. In: Rossi, F. (ed.) CP 2003. LNCS, vol. 2833, pp. 899–903. Springer, Heidelberg (2003)
[18] O'Kelly, M., Morton, E.: A quadratic integer program for the location of interacting hub facilities. European Journal of Operational Research 32(3), 393–404 (1987)
[19] Pelta, D., Sancho-Royo, A., Cruz, C., Verdegay, J.L.: Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. Information Sciences 176(13), 1849–1868 (2006)
[20] Rice, J.: The algorithm selection problem. Advances in Computers 15, 65–118 (1976)
[21] Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82 (1997)