# Achieving Memetic Adaptability by means of Agent-based Machine Learning

Giovanni Acampora, *Member, IEEE,* Jose Manuel Cadenas, *Member, IEEE,* Vincenzo Loia, *Senior Member, IEEE,* and Enrique Muñoz, *Member, IEEE,*

***Abstract*—Over recent years, there has been increasing interest of the research community towards evolutionary algorithms, i.e., algorithms that exploit computational models of natural processes to solve complex optimization problems. In spite of their ability to explore promising regions of the search space, they present two major drawbacks: 1) they can take a relatively long time to locate the exact optimum and 2) may sometimes not find the optimum with sufficient precision. Memetic Algorithms are evolutionary algorithms inspired by both Darwinian principles and Dawkins' notion of a meme, able not only to converge to high quality solutions, but also search more efficiently than their conventional evolutionary counterparts. However, memetic approaches are affected by several design issues related to the different choices that can be made to implement them. This paper introduces a multi-agent based memetic algorithm which executes in a parallel way different cooperating optimization strategies in order to solve a given problem's instance in an efficient way. The algorithm adaptation is performed by jointly exploiting a knowledge extraction process together with a decision making framework based on fuzzy methodologies. The effectiveness of our approach is tested in several experiments in which our results are compared with those obtained by non-adaptive memetic algorithms. The superiority of the proposed strategy is manifest in the majority of cases.**

***Index Terms*—Adaptive Memetic Algorithms, Multi-Agent Systems, Data Mining, Fuzzy Logic**

## I. Introduction

Optimization problems have focused the interest of the research community for a long time. For that reason several strategies have been developed to solve them in a reasonable computational time and find solutions with a near optimum quality. Among these strategies, metaheuristics play a fundamental role. Recent literature, e.g. [1][2][3], reveals a wide variety of problems and methods that appear within this topic, being one of the most studied Evolutionary Algorithms (EAs).

EAs are an interdisciplinary research field which takes its inspiration from natural selection and survival of the fittest. EAs operate on a population of potential solutions by applying the principle of survival of the fittest in order to produce better and better approximations to a sub-optimal solution. At each generation, a new set of approximations is created by selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural genetics. This process leads to the evolution of individuals that are better suited to their environment than the individuals that they were created from.

Nevertheless, although these algorithms have been used to solve complex NP-complete problems [4], it is now well established that they are not well suited to fine tuning search in complex combinatorial spaces and, consequently, the hybridization with other techniques may greatly improve their efficiency [5],[6]. Memetic Algorithms (MAs) are an extension of EAs that apply separate local optimization processes (hill climbing, simulated annealing, tabu search, etc.) to refine individuals. These methods are inspired by models of adaptation that combine the evolutionary adaptation of a population with individual learning of its members. The choice of name is inspired by Dawkins' concept of a meme, which represents a unit of cultural evolution that can exhibit local refinement [7]. In the context of optimization, a meme represents a learning or development strategy. Thus, a MA exhibits the plasticity of individuals that a genetic model fails to capture.

Our idea is to introduce a parallel cooperative adaptive memetic algorithm able to decide, depending on the instance being solved, how to combine and configure different evolutionary and local search metaheuristics to improve overall performance. In detail, the proposed strategy computes two sequential steps dealing respectively with evolutionary and local methodologies. During the first step a collection of population based methods cooperates to find a high quality solution that is forwarded to the second step where a collection of local search methods cooperates to improve it. The cooperation is performed by exchanging solutions among metaheuristics in precise moments and under certain conditions that are controlled by a set of TSK fuzzy rules [26]. The fuzzy engine uses knowledge obtained by a preliminary machine learning process that analyzes the performances of different optimization methods applied to well-defined problem's instances.

The parallel and distributed nature of our approach is fully suitable to be modeled using a multi-agent system where software agents compute metaheuristics under the supervision of a coordinator agent whose intelligence is given from the aforementioned fuzzy rules and machine learning knowledge.

The paper is organized as follows, in Section II we present some related work, in Section III we describe the adaptive memetic algorithm and its developing process, next, in Sec-

The authors thank the MICINN of Spain and the "Fondo Europeo de Desarrollo Regional" (FEDER) for their support under the project TIN2008-06872-C04-03. Thanks also to "Fundación Séneca" for the Funding Program for Research Groups of Excellence (04552/GERM/06) and the support given to E. Muñoz by FPI scholarship program.

G. Acampora and V. Loia are with the Department of Computer Science, University of Salerno, 84084, Fisciano, Salerno, Italy. e-mail: {gacampora, loia}@unisa.it

J. M. Cadenas is with the Dept. of Information and Communications Engineering, University of Murcia, 30071, Murcia, Spain. e-mail: jcadenas@um.es

E. Muñoz is with the European Centre for Soft Computing, 33600, Mieres, Asturias, Spain. e-mail: enrique.munoz@softcomputing.es

tion IV, we introduce a study case, Symple Plant Location Problem, and produce an adaptive memetic algorithm to solve it. In Section V we test the validity of the approach and finally, in section VI we present the conclusions and future work.

## II. Related Work

Memetic algorithms are metaheuristics designed to find solutions to complex and difficult optimization problems [28]. They are extensions of evolutionary algorithms that include a stage of local search optimization as part of their search strategy. MAs have arise as a response to the problems showed by EAs, which generally suffer from slow convergence to locate a precise enough solution because of their failure to exploit local information. This often limits the practicality of EAs on many large-scale real world problems where the computational time is a crucial consideration.

From an optimization point of view [25], MAs have been shown to be both more efficient (i.e., requiring orders of magnitude fewer evaluations to find optima) and more effective (i.e., identifying higher quality solutions) than traditional EAs for some problem domains. As a result, MAs are gaining wide acceptance, in particular, in well-known combinatorial optimization problems where large instances have been solved to optimality and where other metaheuristics have failed to produce comparable results

However, despite the interesting results achieved by MAs, the process of designing effective and efficient MAs still shows some drawbacks. For instance the difficulty of fine tuning their control parameters, which may require extensive tests, and specifically, of finding a problem-specific meme that suits the problem of interest [34]. In fact the choice of memes has been shown to greatly influence the search performance of MAs [9], [13], [20], [24], [29], [33], [35]. This evidence has led research community to develop MAs capable of adapting their behavior to the characteristics of the instance being solved, obtaining third generation MAs or adaptive MAs.

From the different techniques included in adaptive MAs three approaches stand out on account of their results and popularity, namely hyperheuristics, co-evolution of memes and meta-Lamarckian learning. Hyperheuristics [13], [24] follow the idea of fusing a number of different memes together, so that the actual meme applied may differ at each decision point. Co-evolution of memes [33] introduces the idea of including in the representation of each individual information about what meme has to be used to perform local search in the neighborhood of the solution. And meta-Lamarckian learning [29] proposes adapting MA's behavior by on line choosing multiple memes during an MA search in the spirit of Lamarckian learning.

In this paper we propose an adaptive MA diverse from the previously explained, which uses knowledge automatically extracted from precedent executions of the metaheuristics in order to decide how to combine them to obtain better results.

## III. A Parallel Cooperative Adaptive Memetic Algorithm

In this section a Parallel Cooperative Adaptive Memetic Algorithm (PCAMA), based on a multi-agent architecture (see Fig. 1) is introduced. The proposed memetic algorithm is computed in two steps, dealing respectively with evolutionary and local methodologies. In the first step, a collection of different evolutionary optimization strategies (Genetic Algorithm, Particle Swarm Optimization, ...) are executed in a *parallel* way in order to locate the best region of problem's search space. Successively a set of local search strategies (Tabu Search, Simulated Annealing, ...) simultaneously exploits this region in order to find a high quality solution. In each step, the optimization strategies *intelligently choose* their configuration parameters and *cooperate* by exchanging their solutions in *adaptive* way. The adaptability is achieved by means of a fuzzy rule base able to evaluate information extracted by a preliminary machine learning approach [10] that ranks computational performances related to evolutionary and local metaheuristics applied to a given optimization problem.

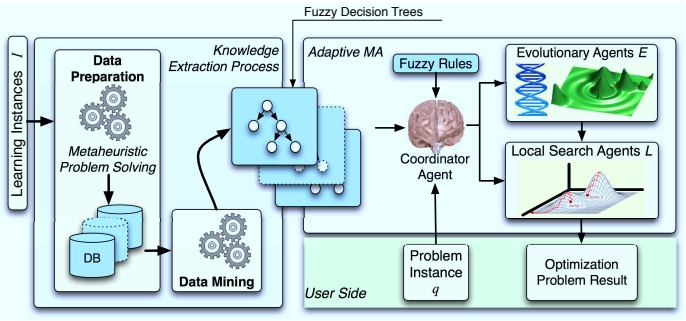


Fig. 1. Adaptive Memetic Architecture.

As previously mentioned the architecture is based on a multi-agent system where a set of so called *optimization agents* computes the cooperating metaheuristics under the supervision of a *coordinator agent* whose intelligence is provided by the aforementioned fuzzy rules and machine learning approach.

### A. Agent-based Memetic Optimization

Let $P$ be an optimization problem and $q$ a given instance of $P$. As said above, the proposed memetic algorithm tries to solve $q$ by means of a collection $M = E \cup L$ of optimization strategies, where $E = \{m_1, m_2, \ldots\}$ and $L = \{m_{|E|+1}, m_{|E|+2}, \ldots, m_{|M|}\}$ contain, respectively, $|E|$ evolutionary strategies and $|L|$ local search methods. The computation of metaheuristics in $M$ is accomplished by two sequential phases dealing respectively with $E$ and $L$. Both evolutionary and local strategies are computed through agent paradigm and, in detail, by exploiting an agents collection $A = \{a_1, a_2, \ldots, a_{|M|}\}$, where the agents subset $\{a_1, a_2, \ldots, a_{|E|}\}$ is related to evolutionary strategies, whereas, the agents subset $\{a_{|E|+1}, a_{|E|+2}, \ldots, a_{|M|}\}$ is related to local optimization methods. The coordinator agent $a^c$ is responsible of initiating optimization process by parallel activating optimization agents and, in each phase, it coordinates the cooperation among optimization agents by choosing their configuration parameters and exchanging their solutions in adaptive way.

The adaptability of $a^c$ is performed by considering knowledge coming from machine learning (see section III-B) coded by means of a collection of fuzzy decision trees, named $T$.

Each branch from root to leaves of trees in $T$ corresponds to the characterization of a particular class of problem instances, whereas, leaves belonging to these branches contain information about suitability of strategies to solve this class. Coordinator agent analyzes these trees in order to individuate the collection of branches more similar to problem's instance $q$. In particular, trees analysis supports coordinator agent to 1) choose the best metaheuristic parameters and 2) rank the metaheuristics suitability to solve $q$ through numerical weights in the range $[0, 1]$. Through weights analysis, $a^c$ may realize that a metaheuristic is poorly performing and, consequently, it may update its solutions by adding a set of better solutions computed by other more suitable metaheuristics.

The cooperation is performed in a synchronous way, i.e. optimization agents evaluate, $n$ times, the objective function then they stop their computation in order to allow the coordinator agent to collect their information and make decisions. After that, each optimization agent continues exploring the search space but executing coordinator agent's orders.

To perform the communication between $a^c$ and optimization agents, a blackboard architecture will be used. Blackboards are data structures usually used as general communication mechanisms. In our proposal each agent has an assigned space on the blackboard where it periodically *writes* information about its found solution. The coordinator *reads* this information and directs the search of each agent.

Fig. 2 shows the architecture of the proposed MA, while Algotrithms 1 and 2 show respectively the pseudocode of optimization and coordinator agents.

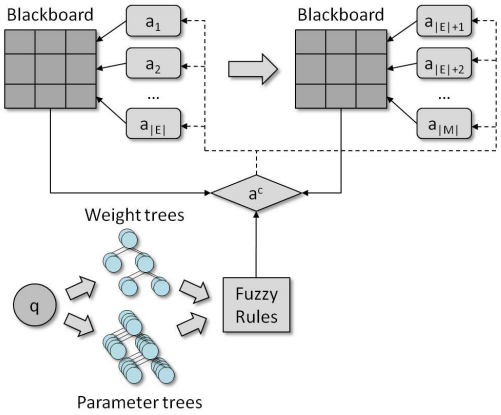


Fig. 2. Adaptive MA architechture.

**Algorithm 1**: Optimization Agent's Pseudocode.

```
Input: q: problem instance, m_i: metaheuristic
       assigned to agent, n: execution time
begin
    read message mess_c from coordinator agent a^c;
    if mess_c is received then
        replace poor solutions with solutions
        contained in mess_c;
    end
    solution = compute metaheuristic m_i
    evaluating objective function n times;
    write solution on blackboard;
end
```

**Algorithm 2**: Coordinator Agent's Pseudocode

```
Input: T: set of trees, n: execution time for each
       a_i, q: problem instance, M = E ∪ L: set of
       metaheuristics
Output: solution: best solution found
begin
    analyze T in order to select the best
    parameter values for each metaheuristic in M;
    while end condition is not satisfied do
        compute each a_i related to E in a parallel
        way;
        exploit fuzzy rules, T and blackboard data
        to select POOR, a set of agents
        performing poorly;
        foreach agent a in POOR do
            compose a message mess_a containing a
            collection of more suitable solutions;
            send mess_a to a;
        end
    end
    bestsolution = read the best solution from
    blackboard;
    use bestsolution as initial solution for agents
    related to L;
    while end condition is not satisfied do
        compute each a_i related to L in a parallel
        way;
        exploit fuzzy rules, T and blackboard data
        to select POOR, a set of agents
        performing poorly;
        foreach agent a in POOR do
            compose a message mess_a containing a
            collection of more suitable solutions;
            send mess_a to a;
        end
    end
    return best solution obtained;
end
```

A crucial question of the proposed architecture is how to model the $a^c$'s intelligence to allow it to evaluate metaheuristics performances and control the cooperation among optimization agents. This goal is achieved using a set of fuzzy rules that exploit the knowledge obtained by machine learning. More precisely, $a^c$ uses fuzzy inference to:

- choose the best metaheuristics' parameters values;
- individuate when poor evolutionary metaheuristics have to receive a collection of individuals from other strategies that are showing a better behavior.
- individuate when local search methods have to recieve solutions from other strategies showing a better behavior.

Coordinator agent implements first behavior by using a collection of $|M|$ fuzzy decision trees obtained through machine learning. These trees analyze $q$ in order to infer - via the inference engine proposed on [21] - the most suitable parameters values for each metaheuristic in $M$. Moreover, the coordinator agent realizes the remaining two behaviors by considering two additional trees coming from machine learning. In detail, these trees analyze $q$ and respectively rank metaheuristics in $E$ and $L$, by returning a weights collection $\Omega = \{\omega_i, i = 1, \ldots, |M|\}$ where $\omega_i \in [0, 1]$ and $\sum_{i=1}^{|E|} \omega_i = \sum_{i=|E|+1}^{|M|} \omega_i = 1$; the weights $\omega_i$, with $i = 1, \ldots, |E|$, are the related to $E$, whereas, the weights $\omega_i$ with $i = |E| + 1, \ldots, |M|$ are the related to $L$.

In short, each weight $\omega_i$ is associated with a metaheuristic $m_i \in M$ and represents its suitability to solve $q$. More precisely, let $m_i, m_j \in E$ or $m_i, m_j \in L$ be two optimization

strategies then $\omega_i > \omega_j$ implies that, according to previous executions, $i^{th}$ metaheuristic obtains an overall better performance than that obtained by $j^{th}$ metaheuristic.

In each phase, the coordinator agent uses a collection of TSK fuzzy rules ($|E|-1$ for each metaheuristic) exploiting $\Omega$ in order to derive the collection of poor strategies. For the sake of simplicity the rules shown below are related to a strategy $m_h \in E$ computed during the evolutionary phase (however rules acting in local phase are equivalent):

**if** $(\omega_1 \cdot d_1)$ is *enough* **then** $poorness_1 = 1$
...
**if** $(\omega_{h-1} \cdot d_{h-1})$ is *enough* **then** $poorness_{h-1} = 1$
**if** $(\omega_{h+1} \cdot d_{h+1})$ is *enough* **then** $poorness_{h+1} = 1$
...
**if** $(\omega_{|E|} \cdot d_{|E|})$ is *enough* **then** $poorness_{|E|} = 1$

where:

- $m_h$ is the metaheuristic being evaluated by the rule;
- $d_i = (\xi(m_i) - \xi(m_h))/\max(\xi(m_i), \xi(m_h))$, where $\xi$ is a measure of performance defined by the user;
- $\omega_i \in [0,1]$ is the weight of $m_i$;
- $enough$ is a fuzzy set with trapezoidal membership function defined by a quadruplet $(a,b,c,d)$ and whose universe of discourse is the range $[0,1]$.
- $poorness_i$ with $i = 1,\ldots,|E|$, is a TSK variable belonging to $[0,1]$. Higher values of $poorness_i$ individuate a bad behavior of $m_i$. In this case, $m_i$ is candidate to change its solution.

The main goal of each rule is to change the position in the search space of a metaheuristic which is showing a bad performance for a position near the solution of another metaheuristic with a better behavior. Note that this rule tries to solve the problem that appears in parallel strategies [14], where unrestricted exchange of solutions favors a prematurely converge to local optimums characterized by a low quality.

To perform the firing of the fuzzy rules a threshold value $\alpha$ is used. In other words, only those rules whose activation exceeds $\alpha$ are fired. In the event that more than one rule is activated then $a^c$ applies all of them. In this way, it is able to simultaneously adapt the behavior of several metaheuristics.

The metaheuristics' solution changing is performed by $a^c$ by taking into account two different situations: 1) a single antecedent clause of rule is fired or 2) many clauses are fired during inference process. In detail, let $S \subset M$ be the collection of strategies related to fired clauses and let $m_h$ be the poor metaheuristic that have to receive better solutions. If a single clause is fired then $|S| = 1$ otherwise $|S| > 1$.

Then, during the evolutionary phase ($m_h \in E$ and $S \subset E$), solutions exchange is performed as follows:

1) $|S| = 1$. A proportion of the worst individuals of $m_h$ is substituted by a set of solutions consisting of the best members of the population of strategy $m_k \in S$. The proportion is equals to $\omega_k$, i.e. the suitability of $m_k$.
2) $|S| > 1$. A proportion of the worst individuals of $m_h$, equals to $\sum_{i=1}^{|S|} \omega_i$, is replaced by a set of solutions chosen by each metaheuristic using previous process.

In contrast, during local optimization phase ($m_h \in L$ and $S \subset L$) the exchange is performed as follows:

- A solution "near" to the best solution obtained among the meta-heuristics $m_k \in S$ is sent to $m_h$. Where "near" means that the best solution is changed by applying, $\lceil \frac{1}{(2 \cdot \omega_k)} \rceil$ times, a mutation operator that depends on $P$.

**Example: Fuzzy Rules Firing Process** In this example we suppose a minimization problem, two evolutionary metaheuristics, GA and PSO ($|E| = 2$), the fuzzy set $enough$ takes the values $[0, 0.1, 1, 1]$ and the $\alpha$ threshold is set to 0.5. Table I shows three different configuration of the framework.

TABLE I
SYSTEM'S STATE

| | GA | PSO | GA | PSO | GA | PSO |
|---|---|---|---|---|---|---|
| weight | 0.43 | 0.57 | 0.68 | 0.32 | 0.55 | 0.45 |
| fitness | 178 | 154 | 178 | 154 | 178 | 154 |
| activation | 0.77 | -0.58 | 0.43 | -0.91 | 0.61 | -0.74 |
| change | yes | no | no | no | yes | no |

In the first case, according to the weights obtained from the tree, PSO has a better overall performance (a higher weight than GA) and also a better solution. When $a^c$ evaluates the rule for GA, it obtains an activation value of $0.77$ ($> \alpha$), which means that PSO has to send some individuals to GA. On the other hand, the rule for PSO is not fired. In the second example, GA has a better overall performance but the current solutions remain the same. In this case, the rule is not fired for GA, although its current solution is worse than that of PSO. Indeed, according to the weights, GA is much better than PSO for this specific instance. In this way, a premature convergence to local solutions of low quality is avoided. The last example is similar to the previous one but the difference of weights is not as high, and thus the rule is fired for GA, because we are more confident of the performance of PSO. ■

### *B. The Knowledge Extraction Process*

As previously mentioned the adaptability of the proposed MA is provided by the knowledge obtained through an extraction process formally introduced in this section. Without loss of generality, let $P$ be a maximization problem, $I$ a training instances set of $P$ characterized by different problem features $F = \{f_1, f_2, \ldots, f_o\}$, and $M = E \cup L$ a collection of $|E|$ evolutionary strategies and $|L|$ local search optimization methods. The aim of the knowledge extraction process is to evaluate the performances of metaheuristics in $M$ when applied to instances in $I$. This process returns a set $T = \{t_1, t_2, \ldots, t_{|M|}\} \cup \{t^E, t^L\}$ containing $|M| + 2$ fuzzy decision trees: $t^E$ and $t^L$ respectively model the suitability of the strategies in $E$ and $L$ by providing the aforementioned weights $\Omega = \{\omega_i, i = 1,\ldots,|M|\}$ and, on the other hand, each tree $t_i$, with $i = 1,\ldots,|M|$, provides the most suitable parameters choice for metaheuristic $m_i$.

The Knowledge extraction process is subdivided in two subphases, Fig. 1, Data Preparation and Data Mining.

*1) Data Preparation:* In this phase, metaheuristics in $M$ are applied to instances in $I$ in order to build a collection of databases $DB$, named refined databases. The obtaining of these databases is carried out in two steps. First, information about the performance of metaheuristics in $M$ is stored in a

set of so called *raw databases*. Successively, data contained in raw databases are processed to compute $DB$, which contains additional information useful to extract significant knowledge during data mining.

In particular, the obtaining of the raw databases is carried out in the following way. Let $m_i$ be a generic metaheuristic in $M$ whose computation depends upon a collection of parameters $P_i = \{p_i^1, p_i^2, \ldots, p_i^{n_i}\}$ and let $V_{i,j} = \{v_{i,j}^1, v_{i,j}^2, \ldots, v_{i,j}^{s_{i,j}}\}$, with $j = 1, \ldots, n_i$, be a finite collection of $s_{i,j}$ suitable values for $p_i^j$, the $j^{th}$ parameter of $i^{th}$ metaheuristic. Consequently, if $c_i = \prod_{l=1}^{n_i} |V_{i,l}|$ is the number of combinations of parameters values (i.e. the cardinality of product set $V_{i,1} \times V_{i,2} \times \ldots \times V_{i,n_i}$ ) then data preparation phase solves each instance $q \in I$ by applying, $r = c_i \cdot k$ times, the metaheuristic $m_i$; $k$ is the *iterative factor*, i.e., a predefined value used to obtain more accurate performance estimations. These performance estimations are utilized to fill $r \cdot |I|$ rows into the database related to metaheuristic $m_i$. In particular, a raw database row contains:

- a description of the specific instance of the problem;
- the values of the parameters used by each metaheuristic;
- the final solution obtained.

Table II shows database entries related to $m_i$, and the following example illustrates the application of this step.

**Example** Let $m_i$ be a GA then parameters sets could be:

$$P_i = \{p_i^1, p_i^2, p_i^3 | p_i^1, p_i^2 \in [0,1] \text{ and } p_i^3 \in \mathbb{N}\}$$
$$V_{i,1} = \{0.1, 0.3, 0.7\}$$
$$V_{i,2} = \{0.01, 0.1, 0.3\}$$

where $p_i^1$ could be the crossover probability and $p_i^2$ the mutation probability. Then, the number of parameters values combinations will be $|V_{i,1}| \cdot |V_{i,2}| = 9$. Now, if $k = 10$ then data preparation process solves each instance $q \in I$ by applying, $k \cdot 9 = 90$ times, the genetic algorithm. The collected data fill $90 \cdot |I|$ rows in genetic algorithm's raw database. ■

Next step is to analyze and refine raw databases in order to compute $DB = \{d_{m_1}, d_{m_2}, \ldots, d_{m_{|M|}}, d_E, d_L\}$ where $|DB| = |M| + 2$. Refined databases are smaller than raw databases and contain additional information useful to extract significant knowledge during data mining. In detail, each $d_{m_i}$ contains information about the most appropriate combinations of parameters values used by $m_i$ to solve instances in $I$. On the other hand, $d_E$ and $d_L$ respectively contain information about the weights $\Omega$ associated to metaheuristics in $E$ and $L$.

In order to build databases in $\{d_{m_i} | i = 1, \ldots, |M|\}$, for each metaheuristic $m_i \in M$ its raw database is processed as follows:

1) for each instance $q_a \in I$ and for each parameters combination $pc_b^i \in V_{i,1} \times V_{i,2} \times \ldots \times V_i^{n_i}$, $b = 1, \ldots, c_i$, the average fitness value is calculated:
$$avg_b^{a,i} = \frac{\sum_{u=1+(b-1)\cdot k}^{b \cdot k} (fit_u^a)}{k}$$
2) for each instance $q_a \in I$, let us define $pc_{best}^{a,i} \in \{pc_b^i | b = 1, \ldots, c_i\}$ as the parameter combination computing the best average fitness value, $\max_{b=1}^{c_i} avg_b^{a,i}$;
3) refined database $d_{m_i}$ is updated with a new entry containing the description of instance $q_a$ and the best parameter combination, i.e, $(f_1^a, f_2^a, \ldots, f_o^a)$ and $pc_{best}^{a,i}$.

The remaining two databases are directly related to evolutionary strategies in $E$ and local search methods in $L$. In detail, the refined database $d_E$ is processed as follows:

1) for each metaheuristic $m_i \in E$, for each instance $q_a \in I$ and for each parameters combination $pc_b^i \in V_{i,1} \times V_{i,2} \times \ldots \times V_{i,n_i}$, the average fitness value is computed:
$$avg_b^{a,i} = \frac{\sum_{u=1+(b-1)\cdot k}^{b \cdot k} (fit_u^a)}{k}$$
2) for each $m_i \in E$ and $q_a \in I$, let us select the best average fitness value $fit_{best}^{a,i} = \max_{u=1+(b-1)\cdot k}^{b \cdot k} (fit_u^a)$ and compute the *metaheuristic suitability weight* as:
$$\omega_i = \frac{fit_{best}^{a,i}}{\sum_{l=1}^{|E|} fit_{best}^{a,l}}$$
3) for each $m_i \in E$ and for each instance $q_a \in I$ the refined database is updated with a new entry containing the description of $q_a$ and the metaheuristic suitability weight, i.e, $(f_1^a, f_2^a, \ldots, f_o^a)$ and $\omega_i$.

Database $d_L$ is analogously built by replacing evolutionary strategies in $E$ with local search methods in $L$. Table III shows refinement processes.

Refined databases will favor the application of data mining phase in computational efficient way. Indeed, Data Preparation applies metaheuristics in $M$ in order to create a collection of $|M|$ raw databases containing $|I| \cdot c_i \cdot k$ rows. The refinement step builds $|M| + 2$ refined databases where databases in the subset $\{d_{m_i} | i = 1, \ldots, |M|\}$ contain $|I|$ entries, whereas, databases $d_L$ and $d_E$ contain, respectively, $|I| \cdot |L|$ and $|I| \cdot |E|$ entries. Consequently, a generic data mining technique analyzes $\sum_{i=1}^{|M|} |I| \cdot c_i \cdot k$ data samples to extract knowledge from raw database, whereas, the same technique analyzes $2 \cdot |M| \cdot |I|$ data samples to extract knowledge from refined databases. Because in real cases $c_i \cdot k >> 2 \Rightarrow 2 \cdot |M| \cdot |I| < \sum_{i=1}^{|M|} |I| \cdot c_i \cdot k$, our data mining approach, based on refined databases, is more efficient than other approaches working on raw data.

*2) Data Mining:* Once gathered performance information and collected it through refined databases $DB$, the data mining phase extracts the collection of knowledge models $T$.

Our data mining approach exploits fuzzy decision trees (FDT) [21] to extract knowledge from $DB$ and build the collection of trees $T = \{t_1, t_2, \ldots, t_{|M|}\} \cup \{t^E, t^L\}$. Two main reasons have guided this choice. First FDT interpretability, decision trees stand out on account of their simplicity and readability, which enables a full understanding of the knowledge that guides $a^c$. Second, their fuzziness, which can improve inference performance by means of approximate reasoning and provide an easy way to obtain the weights $\Omega$.

Before continuing we should explaing FDT construction and inference processes. FDT construction considers instances from a learning database. Each instance is composed by a collection of attributes $A$ where a given attribute, $c \in A$, represents the *class* i.e. an attribute used to locate instances in groups. Then, an FDT codes a constraints collection that

TABLE II

DATABASE RELATED TO METAHEURISTIC $m_i$. FOR EACH INSTANCE $q \in I$, CHARACTERIZED BY ATTRIBUTES $\{f_1, f_2, \ldots, f_o\}$, $m_i$ IS APPLIED $k$ TIMES FOR EACH COMBINATION OF PARAMETER VALUES RELATED TO $\{p_i^1, p_i^2, \ldots, p_i^{n_i}\}$. $fit_z^u$ REPRESENTS THE $z^{th}$ FITNESS VALUE OBTAINED BY APPLYING $m_i$ TO $u^{th}$ INSTANCE IN $I$.

| | | | **Attributes** | | | | **Parameters** | | | | **Final** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $f_1$ | $f_2$ | ... | $f_o$ | $p_i^1$ | $p_i^2$ | ... | $p_i^{n_i}$ | **Solution** |
| Instance | Par. Comb. | It. | | | | | | | | | |
| $q_1$ | 1 | 1 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_1^1$ |
| | | 2 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_2^1$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_k^1$ |
| | 2 | 1 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+1}^1$ |
| | | 2 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+2}^1$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+k}^1$ |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | $c_i$ | 1 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+1}^1$ |
| | | 2 | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+2}^1$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^1$ | $f_2^1$ | ... | $f_o^1$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{c_i\cdot k}^1$ |
| $q_2$ | 1 | 1 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_1^2$ |
| | | 2 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_2^2$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_k^2$ |
| | 2 | 1 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+1}^2$ |
| | | 2 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+2}^2$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+k}^2$ |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | $c_i$ | 1 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+1}^2$ |
| | | 2 | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+2}^2$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^2$ | $f_2^2$ | ... | $f_o^2$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{c_i\cdot k}^2$ |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $q_{\|I\|}$ | 1 | 1 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_1^{\|I\|}$ |
| | | 2 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_2^{\|I\|}$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_k^{\|I\|}$ |
| | 2 | 1 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+1}^{\|I\|}$ |
| | | 2 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+2}^{\|I\|}$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+k}^{\|I\|}$ |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | $c_i$ | 1 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+1}^{\|I\|}$ |
| | | 2 | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+2}^{\|I\|}$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^{\|I\|}$ | $f_2^{\|I\|}$ | ... | $f_o^{\|I\|}$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{c_i\cdot k}^{\|I\|}$ |
| | | | Raw Database Entries | | | | | | | | |

individuates groups defined by $c$. Learning process is shown in Algorithm 3 where the stop condition is defined by the first condition satisfied among the following: (1) the algorithm generates a pure node - a node where all instances are from the same class -, (2) $M$ is empty, (3) the algorithm generates a node containing the minimum number of instances allowed. Each leaf $l$ stores, for each class $c$, the number of instances, $\nu_c^l$, that satisfy its branch's constraints (totally or partially) during learning process. FDTs can deal with both numerical and nominal attributes, but when treating numerical values a previous fuzzy partition is needed. This fuzzy partition can be given by the user or obtained by means of an automatic fuzzy partition method, which commonly obtains better results. The chosen FDT has associated a fuzzy partition algorithm [22] that will be used to obtain the fuzzy partition.

On the other hand, given an instance $q$, inference is performed in the following way:

1) for each leaf, calculate $q$'s membership $\mu_l(q)$, as the satisfaction degree of its branch's constraints.
2) for each class return a membership value $mv_c(q) = \sum_{l=1}^{number\_leaves} \mu_l(q) \cdot \frac{\nu_c^l}{\sum_{i=1}^{number\_class} \nu_i^l}$.

If only one class is needed, then it should be chosen that

TABLE III
FROM RAW DATABASE TO REFINED DATABASES ENTRIES.

| | | | Attributes | | | | Parameters | | | | Final Solution |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $f_1$ | $f_2$ | ... | $f_o$ | $p_i^1$ | $p_i^2$ | ... | $p_i^{n_i}$ | |
| Instance | Par. Comb. | It. | | | | | | | | | |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | 1 | 1 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_1^a$ |
| | | 2 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_2^a$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $fit_k^a$ |
| | 2 | 1 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+1}^a$ |
| | | 2 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+2}^a$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $q_a$ | | k | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $fit_{k+k}^a$ |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | $c_i$ | 1 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+1}^a$ |
| | | 2 | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{(c_i-1)\cdot k+2}^a$ |
| | | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | | k | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $fit_{c_i\cdot k}^a$ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

↓ ↓ Fitness Average Computation ↓ ↓

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | 1 | avg. | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^1$ | $avg_1^{a,i}$ |
| | 2 | avg. | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^1$ | $v_{i,2}^1$ | ... | $v_{i,n_i}^2$ | $avg_2^{a,i}$ |
| $q_a$ | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| | $c_i$ | avg. | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $v_{i,1}^{s_{i,1}}$ | $v_{i,2}^{s_{i,2}}$ | ... | $v_{i,n_i}^{s_{i,n_i}}$ | $avg_{c_i}^{a,i}$ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |

↓ ↓ Database Entry Creation ↓ ↓

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $pc_{best}^{a,i}$ | $f_1^a$ | $f_2^a$ | ... | $f_o^a$ | $\omega_i$ |

Entry of Database $d_{m_i}$ | Entry of Database $d_L$ or $d_E$

**Algorithm 3**: FDT Construction Pseudocode

**Input**: $E$: training instances, *Fuzzy Partition*: attributes partitions in fuzzy sets
**Output**: $t$: FDT obtained
**begin**
  Create root node, containing instances in $E$;
  Obtain $M$, the set of attributes that describe $E$ and where numeric attributes are discretized using *Fuzzy Partition*;
  Nodes_without_expand = {root node};
  **while** *end condition is not satisfied* **do**
    $N$ = next(Nodes_without_expand);
    **foreach** *attribute* $m \in M$ **do**
      Calculate information gain in $M$ taking into account the membership of each instance $e$ to node $N$;
    **end**
    Choose $m_{part}$ the attribute with maximum gain;
    Divide $N$ in subnodes according to the possible outputs of $m_{part}$;
    Add to Nodes_without_expand the subnodes of $N$;
    Eliminate $m_{part}$ of $M$;
  **end**
**end**

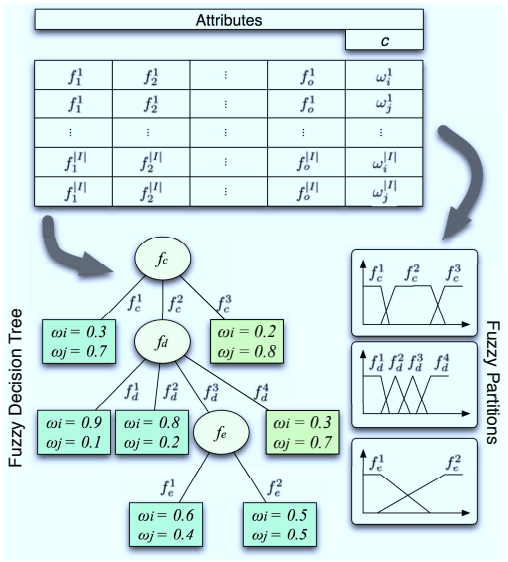


Fig. 3. Data Mining Phase.

with a higher $mv_c(q)$.

In our case, classes are the best parameter combination and the weight and branch constraints are used to analyze instance features. A summary of data mining phase is shown in Fig. 3.

Finally, once this phase is finished, the collection $T$ of FDTs that models $a^c$'s knowledge, is obtained, and the coordinator is ready to control the execution of the optimization agents.

## IV. A Case Study: Simple Plant Location Problem

In order to test and validate the feasibility of our proposal, in this section PCAMA has been applied to a concrete problem: the Simple Plant Location Problem (SPLP).

### A. Simple Plant Location Problem

The simple plant location problem (SPLP) is a well-known combinatorial optimization problem in which some plants or facilities must be chosen among a set of candidates and each of the customers from a given set must be allocated to one of them in such a way that the total cost (location plus allocation) is minimum. Many papers on this problem can be found in the literature (e.g. [11], [12], [17], [18]). SPLP can be formally defined as follows: Consider a set $I = \{1, \ldots, m\}$ of candidate sites for facility location, and a set $J = \{1, \ldots, n\}$ of customers. Each facility $i \in I$ has a fixed cost $f_i$. Every customer $j \in J$ has a demand $b_j$, and $c_{ij}$ is the unit transportation cost from facility $i$ to customer $j$. Without a loss of generality we can normalize the customer demands to $b_j = 1$. It has to be decided: 1) facilities to be established and 2) quantities to be supplied from facility $i$ to customer $j$, such that the total cost is minimized.

Mathematically, the SPLP is formulated as follows:

$$min\left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1} f_i y_i\right)$$

Subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \forall j \in J$$

$$0 \leq x_{ij} \leq y_i \in \{0,1\}, \forall i \in I \text{ and } \forall j \in J$$

where

- $x_{ij}$ represents the quantity supplied from facility $i$ to customer $j$;
- $y_i$ indicates whether facility $i$ is established or not.

Let the set of established facilities be $E_F = \{i | y_i = 1\}$ with cardinality $e = |E_F|$.

As far as computational complexity is concerned, although some special cases of SPLP are solvable in polynomial time, [8], [15], [19], [32], in general, it is a NP-hard problem [23].

In order to apply PCAMA on SPLP, it is crucial to deal with a large number of problem's instances characterized by different features. The choice of problem's instances greatly influences the behaviour of the knowledge extraction process and, consequently, PCAMA's performance. For that reason we have identified some instance's features that may influence the behavior and performance of different solving strategies:

- Number of customers.
- Ratio between facilities and customers.
- Equal or different establishment costs for facilities.
- Connection percentage between facilities and customers.
- Uniform or normal distribution for unit transportation costs.

Through this characterization different SPLP's instances, that highlight distinct problem features.

### B. Construction of the Memetic Algorithm

Hereafter PCAMA configuration process is presented.

*1) Metaheuristics chosen:* A crucial issue in the design of PCAMA is the choice of the metaheuristics that will made up the system. For the problem being considered we have chosen four strategies, two evolutionary metaheuristics, a Genetic Algorithm (GA) and a Particle Swarm Optimization (PSO), and two local search metaheuristics, a Tabu Search (TS) and a Simulated Annealing (SA).

*2) Applying the Knowledge Extraction Process:* The first step of PCAMA is the acquisition of the knowledge that will support coordinator's decisions by applying the aforementioned knowledge extraction process.

The first phase, Data Preparation, required the generation of 192 instances varying on the previously mentioned SPLP's features. Each generated instance was solved 5 times by using GA, PSO, TS and SA with different parameters combinations. Regarding GA different alternatives were chosen for selection strategy, mutation probability and cross probability. Concerning PSO different values were chosen for topology, and parameters that control global memory and local memory. With regard to TS, different values for tabu list size and long term memory are used. Finally, with respect to SA, different values were used for cooling schedule and percentage of neighbors to be considered. The information obtained from these executions was preprocessed to obtain the refined databases. Portions of the different refined databases can be seen on tables IV,V,VI.

TABLE IV
DATABASE $D_E$ FOR POPULATION-BASED METAHEURISTICS

| custs. | rFC | fixedC | %Conn. | gauss | MH | $\omega$ |
|---|---|---|---|---|---|---|
| 1000 | 0.015 | true | 0.445 | true | GA | 0.32 |
| 1000 | 0.015 | true | 0.445 | true | PSO | 0.68 |
| 10000 | 0.031 | false | 0.003 | false | GA | 0.63 |
| 10000 | 0.031 | false | 0.003 | false | PSO | 0.27 |
| ... | ... | ... | ... | ... | ... | ... |

TABLE V
DATABASE $D_L$ FOR TRAJECTORY-BASED METAHEURISTICS

| custs. | rFC | fixedC | %Conn. | gauss | MH | $\omega$ |
|---|---|---|---|---|---|---|
| 1000 | 0.015 | true | 0.445 | true | TS | 0.56 |
| 1000 | 0.015 | true | 0.445 | true | SA | 0.44 |
| 10000 | 0.031 | false | 0.003 | false | TS | 0.72 |
| 10000 | 0.031 | false | 0.003 | false | SA | 0.28 |
| ... | ... | ... | ... | ... | ... | ... |

TABLE VI
DATABASE $D_{m_i}$ RELATED TO THE GENETIC ALGORITHM PARAMETERS

| custs. | rFC | fixedC | %Conn. | gauss | param1 | param2 |
|---|---|---|---|---|---|---|
| 150 | 1 | false | 0.748 | true | 0.001 | 0.6 |
| 1000 | 1 | true | 0.415 | true | 0.01 | 0.6 |
| ... | ... | ... | ... | ... | ... | ... |

After the obtaining of these databases, Data Mining phase was applied in order to obtain the FDTs that configure the coordinator agent. An example FDT can be seen on Fig. 4

In order to complete the coordinator's intelligence definition it is necessary to define the measure of performance $\xi$ used by the TSK rules, the fuzzy set $enough$ and choose the value

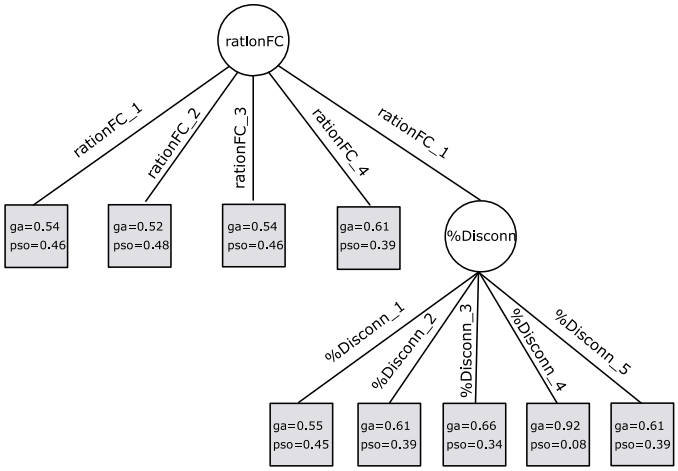


Fig. 4. Example trees.

of the $\alpha$ threshold better suited for the considered problem. For SPLP we have chosen the following options:

- The performance measure, $\xi$, is the value of the SPLP's objective function.
- The fuzzy set $enough$ have a trapezoidal membership function where $(a, b, c, d) = (0, 0.01, 1, 1)$.
- $\alpha$ is set to 0.75.

## V. Computational experiments

In this section we perform some tests to assess the validity of PCAMA by comparing its results with those obtained by the different MAs that can be generated from the sequential combination of the single metaheuristics. Namely:

- Genetic Algorithm + Tabu Search (GA+TS).
- Genetic Algorithm + Simulated Annealing (GA+SA).
- Particle Swarm Optimization + Tabu Search (PSO+TS).
- Particle Swarm Optimization + Simulated Annealing (PSO+SA).

In the following subsections we will explain the configuration of the tests and the results obtained.

### A. Configuration of the experiments

Concerning the system implementation, both coordinator and optimization agents was programmed by using the Java language together with the CILib library [30], [31]. Regarding the fuzzy engine modeling the coordinator's TSK rules, the Fuzzy Markup Language (FML) [27] was exploited.

Tests have been conducted using a test database containing 96 instances that are characterized by different features in terms of number of customers, ratio between facilities and customers, equal or different establishment costs for facilities, connection percentage between facilities and customers, uniform or normal distribution for unit transportation costs.

Both sequential MAs and PCAMA have solved each instance 10 times using 40000 evaluations of the fitness function. In PCAMA the cooperation was performed every 100 evaluations. And every test was executed on an Intel core2 Quad 1.66Ghz with 2GB of Memory.

In order to validate the results obtained, some statistical techniques using non parametric tests [16] have been considered. Specifically, the test used to compare two memetic strategies is Wilcoxon signed-rank test, which is a non-parametric statistical procedure for performing pairwise comparisons between two algorithms analogous to paired t-test. But when more than two methods need to be compared Wilcoxon tests are not enough and a different type of test is needed, in this case we will use Friedman test which is a non-parametric test equivalent to the repeated-measures ANOVA, under the null-hypothesis, it states that the algorithms are equivalent, so a rejection of this hypothesis implies the existence of differences among the performance of all the algorithms studied. After applying Friedman test, a post-hoc test is used to find whether the control algorithm presents statistical differences with regards to the remaining methods. The post-hoc method selected is Benjamin-Hochberger method.

### B. Preliminary tests

In order to graphically show the behavior of the different tested strategies we have carried out some preliminary tests using four instances of the test database. The graphics which illustrate the evolution of the 5 strategies are shown in Fig. 5. These graphics show the average fitness obtained by each strategy on 10 executions of 4000 fitness evaluations.

Different conclusions can be drawn from these graphs. In the first one (Fig. 5(a)), the robustness of the adaptive approach can be appreciated, because although it does not obtain the best results, it computes an average fitness value that is at least similar to that obtained by the best non-adaptive strategy. Indeed PCAMA adaptability is able to exclude SA (that obtained the worst performance) from the problem resolution. In Fig. 5(c), every non-adaptive MA obtains bad performances, whereas PCAMA computes good solutions, probably as a result of its intelligent choice of parameter values. Finally in Fig. 5(b) and 5(d) non-cooperative strategies' behavior is similar but it is outperformed by the one showed by PCAMA. From these results we can anticipate the robustness of the approach, as independently from the results obtained by its components, which may perform poorly, PCAMA always computes high quality solutions, comparable to that obtained by the best non-adaptive MA, being in most cases better.

### C. Comparisons

In this section we present the results obtained from the resolutions of the 96 previously defined SPLP instances. The results are presented in Tables VII, VIII and IX. The first column shows a description of the instance, indicating the number of customers, the ratio between facilities and customers, if the costs of establishing different facilities are constant (C) or different (D), the ratio of disconnection between facilities and customers, and if the unit transportation costs are uniformly (U) or normally distributed (N). The rest of columns indicate the cost of the best solution obtained by each strategy.

The results obtained are very interesting. First, PCAMA obtains better results than every non adaptive MA for the 85.4% of instances. The statistical analysis highlights that

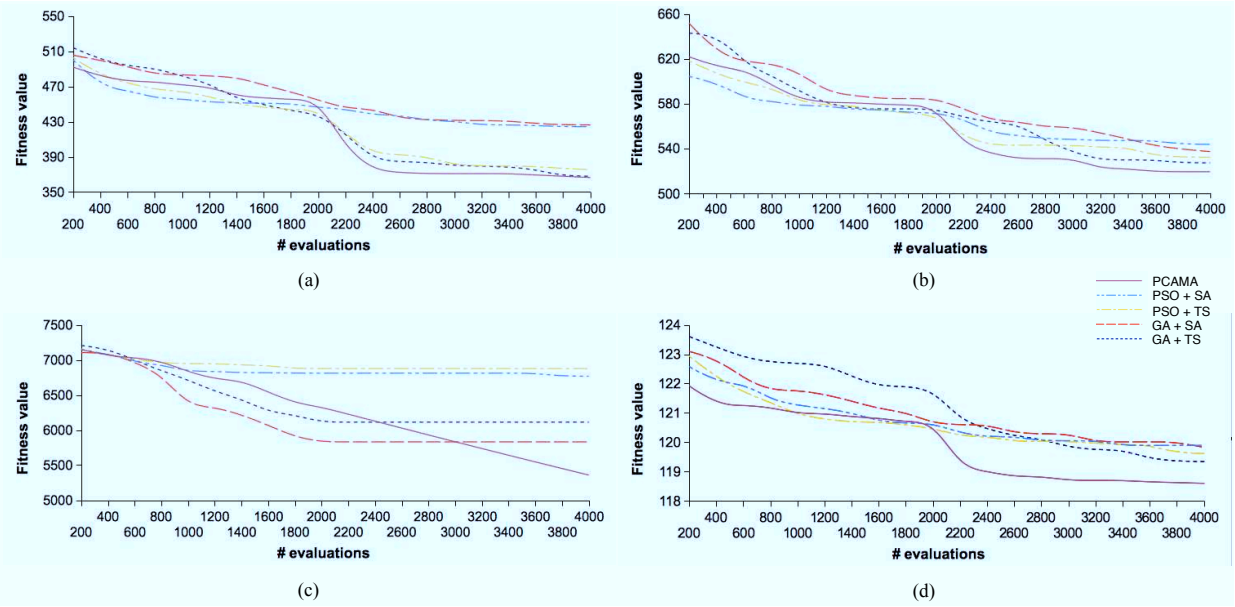


Fig. 5. Evolutions of the different strategies.

TABLE VII
RESULTS (PART I)

| description | GA+TS | GA+SA | PSO+TS | PSO+SA | PCAMA |
|---|---|---|---|---|---|
| 150, 0.33, C, 0.4, N | 19.0 | 25.0 | 19.5 | 30.8 | 18.6 |
| 150, 0.33, C, 0.4, U | 514.3 | 516.2 | 515.9 | 523.0 | 512.3 |
| 150, 0.33, C, 0.5, N | 20.6 | 21.9 | 20.5 | 29.7 | 19.7 |
| 150, 0.33, C, 0.5, U | 757.8 | 758.4 | 762.4 | 772.5 | 757.6 |
| 150, 0.33, C, 0.6, N | 21.2 | 23.7 | 21.6 | 29.1 | 20.4 |
| 150, 0.33, C, 0.6, U | 834.0 | 839.1 | 834.0 | 856.5 | 834.0 |
| 150, 0.33, D, 0.4, N | 77.5 | 137.7 | 75.4 | 481.0 | 70.1 |
| 150, 0.33, D, 0.4, U | 1562.0 | 1631.6 | 1568.7 | 1922.9 | 1546.2 |
| 150, 0.33, D, 0.5, N | 118.6 | 194.0 | 125.9 | 436.1 | 111.1 |
| 150, 0.33, D, 0.5, U | 1556.7 | 1612.7 | 1552.6 | 1818.2 | 1549.5 |
| 150, 0.33, D, 0.6, N | 170.8 | 232.8 | 150.2 | 435.7 | 147.3 |
| 150, 0.33, D, 0.6, U | 2071.8 | 2195.7 | 2073.9 | 2345.6 | 2068.2 |
| 150, 0.66, C, 0.4, N | 39.9 | 41.8 | 41.1 | 50.0 | 15.3 |
| 150, 0.66, C, 0.4, U | 397.1 | 401.7 | 402.7 | 409.8 | 371.5 |
| 150, 0.66, C, 0.5, N | 19.2 | 30.5 | 19.2 | 48.7 | 18.3 |
| 150, 0.66, C, 0.5, U | 398.7 | 422.8 | 398.6 | 431.9 | 396.9 |
| 150, 0.66, C, 0.6, N | 20.7 | 30.0 | 20.5 | 47.9 | 20.0 |
| 150, 0.66, C, 0.6, U | 524.5 | 524.9 | 535.2 | 545.4 | 495.4 |
| 150, 0.66, D, 0.4, N | 48.1 | 610.1 | 41.0 | 1403.0 | 44.1 |
| 150, 0.66, D, 0.4, U | 1170.0 | 1486.2 | 1169.3 | 2072.7 | 1164.4 |
| 150, 0.66, D, 0.5, N | 57.2 | 588.6 | 53.9 | 1568.6 | 55.6 |
| 150, 0.66, D, 0.5, U | 1363.3 | 1753.3 | 1371.0 | 2324.0 | 1350.7 |
| 150, 0.66, D, 0.6, N | 85.1 | 481.9 | 89.6 | 1302.8 | 79.0 |
| 150, 0.66, D, 0.6, U | 1499.9 | 1843.4 | 1496.4 | 2342.0 | 1499.6 |

TABLE VIII
RESULTS (PART II)

| description | GA+TS | GA+SA | PSO+TS | PSO+SA | PCAMA |
|---|---|---|---|---|---|
| 1000, 0.02, C, 0.4, N | 113.4 | 113.6 | 114.2 | 113.9 | 113.1 |
| 1000, 0.02, C, 0.4, U | 7875.3 | 7875.3 | 7875.3 | 7875.3 | 7875.3 |
| 1000, 0.02, C, 0.5, N | 119.2 | 118.8 | 119.9 | 118.9 | 118.0 |
| 1000, 0.02, C, 0.5, U | 9825.8 | 9825.8 | 9825.8 | 9825.8 | 9825.8 |
| 1000, 0.02, C, 0.6, N | 115.8 | 115.3 | 116.2 | 115.7 | 114.5 |
| 1000, 0.02, C, 0.6, U | 13171.9 | 13171.9 | 13171.9 | 13171.9 | 13171.9 |
| 1000, 0.02, D, 0.4, N | 228.2 | 214.3 | 235.4 | 214.3 | 216.3 |
| 1000, 0.02, D, 0.4, U | 8984.2 | 8984.2 | 8984.2 | 8984.2 | 8984.2 |
| 1000, 0.02, D, 0.5, N | 382.4 | 410.2 | 395.6 | 440.8 | 351.0 |
| 1000, 0.02, D, 0.5, U | 11227.6 | 11227.6 | 11227.6 | 11227.6 | 11227.6 |
| 1000, 0.02, D, 0.6, N | 523.8 | 489.9 | 529.8 | 485.3 | 488.7 |
| 1000, 0.02, D, 0.6, U | 13873.1 | 13873.1 | 13873.1 | 13873.1 | 13873.1 |
| 1000, 0.33, C, 0.4, N | 200.2 | 204.9 | 217.5 | 253.2 | 110.9 |
| 1000, 0.33, C, 0.4, U | 1012.3 | 1047.8 | 1009.0 | 1072.2 | 925.7 |
| 1000, 0.33, C, 0.5, N | 199.5 | 205.6 | 203.0 | 249.3 | 115.4 |
| 1000, 0.33, C, 0.5, U | 1211.6 | 1227.6 | 1209.0 | 1260.0 | 1121.1 |
| 1000, 0.33, C, 0.6, N | 235.8 | 237.7 | 258.8 | 256.5 | 117.1 |
| 1000, 0.33, C, 0.6, U | 1353.7 | 1360.6 | 1365.1 | 1417.0 | 1245.9 |
| 1000, 0.33, D, 0.4, N | 3412.9 | 4474.3 | 3516.8 | 6587.4 | 170.8 |
| 1000, 0.33, D, 0.4, U | 4764.6 | 5595.1 | 4760.4 | 7346.9 | 3066.8 |
| 1000, 0.33, D, 0.5, N | 3453.2 | 4302.4 | 3375.9 | 6393.8 | 162.4 |
| 1000, 0.33, D, 0.5, U | 5390.3 | 6580.4 | 5456.8 | 8118.6 | 3661.6 |
| 1000, 0.33, D, 0.6, N | 3702.9 | 4265.4 | 3532.2 | 6526.0 | 168.0 |
| 1000, 0.33, D, 0.6, U | 5877.5 | 6991.0 | 6038.9 | 8326.4 | 4222.1 |
| 1000, 0.66, C, 0.4, N | 340.9 | 343.6 | 407.7 | 411.9 | 252.7 |
| 1000, 0.66, C, 0.4, U | 988.9 | 996.0 | 988.7 | 1046.8 | 850.5 |
| 1000, 0.66, C, 0.5, N | 339.2 | 335.8 | 399.3 | 403.4 | 246.2 |
| 1000, 0.66, C, 0.5, U | 1091.3 | 1104.4 | 1099.7 | 1160.1 | 952.6 |
| 1000, 0.66, C, 0.6, N | 384.5 | 381.5 | 413.5 | 411.9 | 247.4 |
| 1000, 0.66, C, 0.6, U | 1179.7 | 1192.6 | 1200.4 | 1233.4 | 1027.8 |
| 1000, 0.66, D, 0.4, N | 10553.8 | 10547.7 | 13683.3 | 13743.5 | 4085.5 |
| 1000, 0.66, D, 0.4, U | 11689.0 | 11914.2 | 14456.5 | 14912.7 | 5379.3 |
| 1000, 0.66, D, 0.5, N | 10959.2 | 10827.2 | 13599.6 | 14037.5 | 4206.4 |
| 1000, 0.66, D, 0.5, U | 14536.3 | 14412.4 | 16014.2 | 15707.5 | 5776.0 |
| 1000, 0.66, D, 0.6, N | 10753.0 | 10751.8 | 13728.8 | 13616.9 | 3711.6 |
| 1000, 0.66, D, 0.6, U | 11926.4 | 12077.9 | 14801.9 | 15159.4 | 6021.1 |

PCAMA solutions are better than those computed by non-adaptive strategies with a certainty of the 99.9%. Moreover, by comparing PCAMA with a theoretical MA that chooses, for each instance, the best non-adaptive strategy, an average fitness improvement equals to the 14.13% is achieved.

If we focus on the results obtained for instances of 1000 customers, we can observe the highest differences in performance among PCAMA and non-adaptive strategies. This could be due to the difficulty of instances. Indeed, instances of 150 customers can be small and it is easier to obtain good results even for non-adaptive strategies, on the other hand, instances of 10000 customers have a low ratio between facilities and customers, and thus also these instances become easy to solve, and for that reason the results obtained by every strategy are quite similar.

## VI. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a parallel cooperative adaptive memetic algorithm, named PCAMA, aimed to solve optimization problems by deciding, depending on the instance

TABLE IX
RESULTS (PART III)

| description | GA+TS | GA+SA | PSO+TS | PSO+SA | PCAMA |
|---|---|---|---|---|---|
| 10000, 0.01, C, 0.4, N | 1130.4 | 1146.4 | 1128.7 | 1165.0 | 1126.4 |
| 10000, 0.01, C, 0.4, U | 17617.9 | 17617.9 | 17672.2 | 19288.8 | 17617.9 |
| 10000, 0.01, C, 0.5, N | 1135.1 | 1156.8 | 1137.95 | 1177.7 | 1135.5 |
| 10000, 0.01, C, 0.5, U | 21940.9 | 21940.9 | 22007.6 | 24268.1 | 21940.9 |
| 10000, 0.01, C, 0.6, N | 1131.3 | 1150.1 | 1131.8 | 1165.8 | 1126.3 |
| 10000, 0.01, C, 0.6, U | 27926.6 | 27926.6 | 27980.1 | 31426.5 | 27926.6 |
| 10000, 0.01, D, 0.4, N | 1236.9 | 1601.3 | 1215.9 | 2603.6 | 1220.6 |
| 10000, 0.01, D, 0.4, U | 22279.4 | 22279.4 | 23882.4 | 23837.6 | 22279.4 |
| 10000, 0.01, D, 0.5, N | 1282.2 | 1623.8 | 1288.3 | 2430.9 | 1284.6 |
| 10000, 0.01, D, 0.5, U | 26827.2 | 26827.2 | 26920.0 | 29151.5 | 26827.2 |
| 10000, 0.01, D, 0.6, N | 1406.7 | 1800.6 | 1381.2 | 2811.9 | 1413.2 |
| 10000, 0.01, D, 0.6, U | 32700.4 | 32700.4 | 32780.6 | 35157.1 | 32700.4 |
| 10000, 0.02, C, 0.4, N | 1140.8 | 1183.4 | 1139.3 | 1218.1 | 1135.0 |
| 10000, 0.02, C, 0.4, U | 10528.6 | 10528.6 | 10569.2 | 11715.4 | 10528.6 |
| 10000, 0.02, C, 0.5, N | 1146.2 | 1194.0 | 1143.3 | 1227.1 | 1146.7 |
| 10000, 0.02, C, 0.5, U | 12439.2 | 12439.2 | 12511.5 | 14050.3 | 12439.2 |
| 10000, 0.02, C, 0.6, N | 1140.1 | 1186.1 | 1137.5 | 1216.2 | 1141.5 |
| 10000, 0.02, C, 0.6, U | 15110.7 | 15110.7 | 15171.7 | 17219.1 | 15110.7 |
| 10000, 0.02, D, 0.4, N | 1205.7 | 3236.5 | 1199.4 | 4644.3 | 1213.3 |
| 10000, 0.02, D, 0.4, U | 18176.9 | 19292.3 | 18172.0 | 19539.6 | 18168.6 |
| 10000, 0.02, D, 0.5, N | 1221.9 | 3173.5 | 1217.7 | 4614.8 | 1266.3 |
| 10000, 0.02, D, 0.5, U | 21078.4 | 22040.4 | 21089.9 | 22394.8 | 21049.2 |
| 10000, 0.02, D, 0.6, N | 1203.4 | 2952.9 | 1210.1 | 4334.2 | 1234.3 |
| 10000, 0.02, D, 0.6, U | 24515.0 | 25304.7 | 24520.6 | 25813.4 | 24488.0 |
| 10000, 0.03, C, 0.4, N | 1220.9 | 1225.9 | 1220.4 | 1270.1 | 1141.7 |
| 10000, 0.03, C, 0.4, U | 8320.0 | 8320.0 | 9143.2 | 9159.5 | 8314.0 |
| 10000, 0.03, C, 0.5, N | 1218.8 | 1228.2 | 1228.5 | 1269.9 | 1146.1 |
| 10000, 0.03, C, 0.5, U | 9606.7 | 9606.7 | 10724.9 | 10895.5 | 9604.5 |
| 10000, 0.03, C, 0.6, N | 1218.8 | 1228.5 | 1214.9 | 1268.2 | 1138.8 |
| 10000, 0.03, C, 0.6, U | 11325.8 | 11325.8 | 12815.3 | 13030.6 | 11325.8 |
| 10000, 0.03, D, 0.4, N | 3622.1 | 4867.2 | 3510.7 | 6822.6 | 1191.6 |
| 10000, 0.03, D, 0.4, U | 17174.7 | 18111.9 | 17186.9 | 19162.0 | 16000.9 |
| 10000, 0.03, D, 0.5, N | 3695.6 | 4913.6 | 3876.8 | 7040.5 | 1222.1 |
| 10000, 0.03, D, 0.5, U | 20093.4 | 21119.9 | 20068.0 | 22073.8 | 18913.7 |
| 10000, 0.03, D, 0.6, N | 3564.7 | 4841.2 | 3425.2 | 6741.6 | 1278.8 |
| 10000, 0.03, D, 0.6, U | 22353.3 | 23512.1 | 22394.1 | 23951.3 | 21245.0 |

being solved, how to combine and configure different evolutionary and local search metaheuristics. The computation of the proposed strategy is divided in two steps, dealing respectively with evolutionary and local methodologies. In the first step, different evolutionary optimization strategies cooperatively explore the search space in a parallel way. The best solution found by this step is then forwarded to the second step, where a set of local search strategies improves it by means of a parallel cooperative exploration of the search space.

The cooperation among the different strategies is performed by intelligently exchanging solutions in precise moments and under certain conditions. The exchange control is supervised by a set of fuzzy rules, which exploits knowledge modeled by a collection of fuzzy decision trees and obtained using a preliminary learning process, which analyzes the performance of different methods applied to well-defined problem's instances. With this knowledge the fuzzy engine can predict, depending on the instance being solved, the behavior of each metaheuristic, deciding its relevance and choosing a suitable set of parameter values.

The parallel and distributed nature of PCAMA is fully suitable to be modeled using a multi-agent system where software agents compute the cooperating metaheuristics under the supervision of a coordinator agent whose intelligence is given from the aforementioned fuzzy rules and decision trees.

PCAMA is able to operate independently in order to find high quality solutions to new instances of the problem of interest maximizing the fitness and the convergence rate. To test this statement we applied the methodology to a well-known optimization problem, SPLP. First we executed the construction process, showing how each phase was performed, and after that we carried out some computational tests in order to check the effectiveness of the obtained strategy. In these tests the proposed strategy is compared with non adaptive memetic algorithms, obtaining very interesting results, being confirmed its superiority by statistical tests which conclude with a 99.9% of certainty that its results are better.

## REFERENCES


[1] F. Glover, G.A. Kochenberger, Handbook of Metaheuristics, Kluwer Academic, Dordrecht, 2003.

[2] T. M. Chan, K. F. Man, K. S. Tang, S. Kwong, "A Jumping-Genes Paradigm for Optimizing Factory WLAN Network," *Industrial Informatics, IEEE Transactions on*, vol.3, no.1, pp.33-43, Feb. 2007.

[3] Lo, C.H., Fung, E.H.K., Wong, Y.K., , "Intelligent Automatic Fault Detection for Actuator Failures in Aircraft," *Industrial Informatics, IEEE Transactions on*, vol.5, no.1, pp.50-55, Feb. 2009.

[4] Huai-Kuang Tsai, Jinn-Moon Yang, Yuan-Fang Tsai, Cheng-Yan Kao, "An evolutionary algorithm for large traveling salesman problems," *IEEE Trans. Syst. Man Cyb. Part B* , vol.34, no.4, pp.1718-1729, Aug. 2004.

[5] L. Davis, Handbook of Genetic Algorithms. NY: Van Nostrand (1991).

[6] D. Wolpert and W. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput., vol. 1, no. 1, pp. 67–82 (1997).

[7] R. Dawkins, The Selfish Gene. New York: Oxford Univ. Press (1976).

[8] A.A. Aqeev, V.S. Beresnev, Polynomially Solvable Cases of the Simple Plant Location Problem, in Proc. of the First Integer Programming and Combinatorial Optimization Conference (1990) pp. 1–6.

[9] E. K. Burke, G. Kendall, and E. Soubeiga, A tabu search hyperheuristic for timetabling and rostering, J. Heuristics, vol. 9, no. 6 (2003).

[10] J.M. Cadenas, M.C. Garrido, E. Muñoz. Using machine learning in a cooperative hybrid parallel strategy of metaheuristics. *Inform. Sciences*. 179: 3255–3267. 2009.

[11] D.C. Cho, E.L. Johnson, M.W. Padberg, M.R. Rao, On the uncapacitated plant location problem I: valid inequalities and facets, Math. Oper. Res. 8 (4) (1983) 579–589.

[12] G. Cornujols, M.L. Fisher, G.L. Nemhauser, On the uncapacitated location problem, Ann. Discrete Math. 1 (1977) 163–177.

[13] P. Cowling, G. Kendall, and E. Soubeiga, A hyperheuristic approach to scheduling a sales summit, in PATAT 2000, Springer Lecture Notes in Computer Science, Konstanz, Germany, pp. 176–190 (2000).

[14] T. G. Crainic, M. Gendreau, P. Hansen, N. Mladenovic, Cooperative parallel variable neighborhood search for the p-median, J. of Heuristics 10 (2004) 293–314.

[15] C. De Simone, C. Mannino, Easy Instances of the Plant Location Problem, Tech. Rep. R-427. Gennaio, University of Roma, Italy, (1996).

[16] S. García, A. Fernández, J. Luengo, F. Herrera, A study statistical of techniques and performance measures for genetics-based machine learning: accuracy and interpretability, *SoftComputing* 13 (10), 959–977 (2009).

[17] B. Goldengorin, D. Ghosh, G. Siersksma, Branch and peg algorithms for the simple plant location problem, Comput. Oper. Res. 31 (2) (2004) 241–255.

[18] S. Wang, J. Watada, W. Pedrycz, "Value-at-Risk-Based Two-Stage Fuzzy Facility Location Problems," *Industrial Informatics, IEEE Transactions on* , vol.5, no.4, pp.465–482, November 2009.

[19] V.P. Grishukhin, On Polynomial Solvability Conditions for the Simplest Plant Location Problem, in Selected topics in discrete mathematics (1994) pp. 37–46.

[20] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling, IEEE Trans. Evol. Comput., vol. 7, pp. 204–223 (2003).

[21] C.Z. Janikow, Fuzzy decision trees: issues and methods, IEEE Trans. Syst. Man Cyb., Part B. 28(1) (1998) pp. 1–14.

[22] J.M. Cadenas, M.C. Garrido, R. Martínez Una estratégia de particionamiento fuzzy basada en combinación de algoritmos. CAEPIA-TTIA-2009. pp. 379-388. Sevilla. España, 2009.

[23] J. Krarup and P.M. Pruzan, The Simple Plant Location Problem: Survey and Synthesis. European J. Oper. Res. 12 (1983) pp. 36–81.

[24] N. Krasnogor, B. Blackburne, J. D. Hirst, and E. K. N. Burke, Multimeme algorithms for the structure prediction and structure comparison of proteins, Parallel Problem Solving From Nature, LNCS (2002).

[25] N. Krasnogor, J.E. Smith, A tutorial for competent memetic algorithms: Model, taxonomy and design issues. IEEE Trans. Evol. Algorithms 9(5) pp. 474–488 (2005).
[26] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. on Syst. Man Cyb.*, vol. 15, no. 1, pp. 116-132, February 1985.
[27] G. Acampora and V. Loia, Fuzzy control interoperability and scalability for adaptive domotic framework, IEEE Trans. Ind. Inform., vol.1, no.2, pp. 97–111, May 2005
[28] P. Moscato, On evolution, search, optimization, GAs and martial arts: toward memetic algorithms, California Inst. Technol., Pasadena, CA, Tech. Rep. Caltech Concurrent Comput. Prog. Rep. 826 (1989).
[29] Y. S. Ong, A. J. Keane, Meta-Lamarckian in memetic algorithm, IEEE Trans. Evol. Comput., vol. 8, pp. 99110 (2004).
[30] G. Pampará, A.P. Engelbreght, T. Cloete, CIlib: A Collaborative Framework for Computational Intelligence Algorithms Part I in Proc. for IJCNN 2008, Hong Kong (2008) pp. 1751–1758
[31] G. Pampará, A.P. Engelbreght, T. Cloete, CIlib: A Collaborative Framework for Computational Intelligence Algorithms Part II in Proc. for IJCNN 2008, Hong Kong (2008) pp. 1765–1774
[32] C. Ryu and M. Guignard, An Exact Algorithm for the Simple Plant Location Problem with an Aggregate Capacity Constraint, TIMS/ORSA Meeting. Orlando, FL (1992) 92-04-09.
[33] J. E. Smith et al., Co-evolving memetic algorithms: Initial investigations, Parallel Problem Solving From NaturePPSN VII, G. Guervos et al., Eds. Berlin, Germany: Springer, LNCS, pp. 537–548 (2002).
[34] A. Torn and A. Zilinskas, Global Optimization. Berlin, Germany: Springer-Verlag, vol. 350, LNCS (1989).
[35] N. Zhu, Y. S. Ong, K. W. Wong, and K. T. Seow, Using memetic algorithms for fuzzy modeling, Austral. J. Intell. Inform. Process. vol. 8, no. 3, pp. 147–154 (2004).