An Adaptive Multi-Agent Memetic System for Personalizing e-Learning Experiences

Giovanni Acampora*, Matteo Gaeta†, Enrique Muñoz‡ and Autilia Vitiello*
*Dept. of Computer Science, University of Salerno, Fisciano, Italy 84084
Email: {gacampora, loia, avitiello}@unisa.it
†Centro di Ricerca in Matematica Pura ed Applicata, Fisciano, Italy 84084
Email: gaeta@crmpa.unisa.it
‡European Centre for Soft Computing, Mieres, Spain 33600
Email: enriquemuba@um.es

***Abstract*—The rapid changes in modern knowledge, due to exponential growth of information sources, are complicating learners' activity. For this reason, novel approaches are necessary to obtain suitable learning solutions able to generate efficient, personalized and flexible learning experiences. From this point of view, the use of different cooperative intelligent agents can be exploited to analyze learner's preferences and generate high quality learning presentations which provide attractive learning solutions. In particular, to achieve this goal this paper exploits an ontological representation of the learning environment and an adaptive memetic algorithm based on a cooperative multi-agent framework. In this framework different agents analyze the e-learning instance and solve it in a parallel way, cooperating among them. This cooperation is performed by jointly exploiting data mining, via fuzzy decision trees, together with a decision making framework exploiting fuzzy methodologies. As will be shown in the experimental results section, this multi-agent strategy is capable of speeding up the convergence to high-quality personalized e-learning experiences.**

***Index Terms*—Adaptive Memetic Algorithms, Multi-Agent Systems, E-learning.**

I. Introduction

Multi-agent systems have been applied to many different fields obtaining outstanding results. By using intelligent agents we are able to solve complex problems, typically difficult to solve individually, but that can be solved by the interaction of different entities. One of this kind of problems are optimization problems, which are those problems in which the best solution has to be found among a set of feasible solutions. In the context of e-learning, the problem of binding subjects and technologies with learning activities and leaner's preferences can be modeled as an optimization problem, in particular the one called *Learning Presentation Problem* (LPP). For this reason, we believe that a multi-agent framework can help us in its resolution.

Regarding e-learning solutions, for many years most of them have been modeled as learning software platforms which constrain learners to learn following a predefined approach without taking into account their preferences. In addition, they tend to ignore several tools provided by Web 2.0, that could improve learners' experience. In order to face these weaknesses, new e-learning solutions should support the whole learning process, and manage new e-learning 2.0 scenarios [1] by incorporating tools like Blogs, Wikis, Podcast, and so on. Moreover, next generation Learning Management Systems (LMSs) should dynamically and intelligently create personalized e-learning experiences adapted in the Web 2.0 environment.

In this scenario, a representation model to describe both educational domains and learners characteristics is required. Ontologies [2] are a good methodology to model the e-learning contexts and enable a convenient way to bind subjects with learning activities in order to fit specific learning objectives, which we defined as LPP. LPP could be viewed as a modified version of the Uncapacited Facility Location Problem (UFLP) [3] and, consequently, solved through a specific deterministic algorithm. However, this approach is not convenient for Web 2.0 environment characterized by distributed repositories and a wide number of learning activities and learning paths made by numerous subjects.

Starting from this consideration, this work proposes an adaptive multi-agent memetic algorithm which decides, depending on the instance being solved, how to combine and configure different optimization agents in order to derive high-quality learning experiences in the e-learning 2.0 scenarios. In detail, a collection of agents analyzes learner preferences and generates high quality learning presentations by executing, in a parallel way, different cooperating optimization strategies. The metaheuristics cooperation is performed by exchanging solutions among optimization methods in precise moments and under certain conditions that are controlled by a set of fuzzy rules. 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.

As will be shown in the experimental results section, where a plug-in for an open-source LMS has been designed and tested, the proposed strategy is capable of speeding up the convergence to high-quality personalized e-learning experiences.

II. The Ontological e-Learning Environment

Our e-Learning system is based on 1) a set of models able to represent the main entities involved in the process of teaching/learning and on 2) a set of methodologies, leveraging on such models, for the generation of individualized learning experiences with respect to learning objectives, pre-existing

knowledge and learning preferences. The four models adopted by our solution, based on the *Learning Model* [4], are summarized below:

- the *domain model* represents the knowledge domain that is object of teaching by means of concepts, relations among concepts and teaching preferences connected with concepts;
- the *learner model* represents a learner including concepts that he knows as well as his learning preferences;
- the *learning activity model* represents a single learning object or service that may be used as a building block to generate a learning experience;
- the *unit of learning model* represents a whole learning experience personalized for a single learner and composed by a set of target concepts and a sequence of learning activities needed to learn the target concepts.

By exploiting these models, our system generates a personalised *learning path* (sequence of concepts to be taught) for each learner through a *learning path generation algorithm* and introduces placeholders for testing activities.

Once the learning path is available, the system selects a fragment of the learning path and efficiently generates the best *learning presentation* (sequence of learning activities) for each enrolled learner by applying our proposal of Multi-Agent Memetic Algorithm. The learner then undertakes the learning and testing activities of the learning presentation until its end. When the learner ends a presentation fragment, his learner model is updated on the basis of the results of testing activities and a new learning presentation fragment is generated (possibly including recovery activities for concepts that he did not understand).

A. The Domain Model

The domain model describes, in a machine-understandable way, the piece of the educational domain that is relevant for the e-learning experience we want to define, concretize and broadcast. Our approach is design an ontology composed by a set of concepts representing the topics to be taught and a set of relations between concepts representing connections among topics. Such a structure can be formally represented as a $(n+1)$-tuple $G(C, R_1, \ldots, R_n)$ where C is the set of nodes representing domain concepts and each R_i is a set of arcs corresponding to the i^{th} kind of relation. Several kinds of relations are allowed. As an example we can consider a concept graph $G(C, BT, IRB, SO)$ with three relations BT, IRB and SO whose meaning is explained below (where a and b are two concepts of C):

- $BT(a, b)$ means that the concept *a belongs to b*, i.e., b can be understood if every a so that a belongs to b is already learned. (hierarchical relation);
- $IRB(a, b)$ means that the concept *a is required by b*, i.e., a necessary condition to study b is to have understood a (ordering relation);
- $SO(a, b)$ means that the *suggested order* between a and b is that a precedes b, i.e., to favour learning, it is desirable to study a before b (ordering relation).

TABLE I
EXAMPLE OF DIDACTICAL PROPERTIES AND FEASIBLE VALUES.

Properties	**Feasible Values**
didactic method	deductive, inductive, etc.
activity type	text reading, video clip, simulation discussion with a peer, discussion with the teacher, etc.
interactivity level	high, medium, low

A set of *teaching preferences* may be added to the domain model to define feasible teaching strategies that may be applied for each available concept. Such preferences are represented as a function $TP(C \times Props \times PropVals) \rightarrow [0, 10]$ where $Props$ is the set of didactical properties and $PropVals$ is the set of feasible values for such properties. The Table I provides some (non exhaustive) example of didactical property and associated feasible values. It is worth noting that TP is defined only for couples of $Props$ and $PropVals$ elements belonging to the same row in Table I.

B. The Learner Model

The learner is the main actor of the whole learning process and it is represented with a cognitive state and a set of learning preferences. The *cognitive state* represents the knowledge reached by a learner at a given time and it is represented as a function $CS(C) \rightarrow [0, 10]$ where C is the set of concepts of a given domain model. Given a concept c, $CS(c)$ indicates the degree of knowledge (or grade) reached by a given learner for c; the value 0 indicates *no knowledge*, whereas, the value 10 indicates *full knowledge*. If such grade is greater than a given "passing" threshold θ then c is considered as known, otherwise it is considered as unknown.

The *learning preferences* provide an evaluation of learning strategies that may be adopted for a given learner. They are represented as an application $LP(Props \times PropVals) \rightarrow [0, 10]$ where $Props$ and $PropVals$ are the same sets defined in Table I for teaching preferences, whereas, the value 0 and 10, represent, respectively, the lowest and highest level of learning preference. Differently from teaching preferences, learning preferences are not linked to a domain concept but refer to a specific learner. The cognitive state of any learner is initially void (i.e., $CS(c) = 0$ for any c included in a given domain model) and may be initialized on a teaching domain with a pre-test. Learning preferences may be initialized by the teacher or directly by learners through a questionnaire capable of evaluating learners styles and transform them in suitable values for learning preferences. Both parts of the learner model are automatically updated by the system during learning activities.

C. The Learning Activity Model

A *learning activity* must be performed by a learner to acquire one or more domain concepts. Activities may relate to learning objects (e.g., textual lessons, presentations, videoclips, podcasts, simulations, exercises, etc.) or learning services (e.g.,

virtual labs, wikis, folksonomies, forums, etc.). Our system uses learning activities as bricks to generate learning experiences. In order to be used in this way, a learning activity LO is described through the following elements:

- a set of *concepts* C_{LO} related to a given domain model, that are covered by in the learning activity;
- a set of *didactical properties* expressed as an application $DP_{LO}(property) = value$ representing learning strategies applied by the learning activity;
- a set of *cost properties* expressed as an application $CP_{LO}(property) = value$ that must be taken into account in the optimisation process connected with the *learning presentation generation algorithm.*

Didactical properties components have the same meaning with respect to teaching and learning preferences, i.e., property and value may assume values from a closed vocabulary (see table I). Differently from learning and teaching preferences, they are neither linked to a domain concept nor to a specific student but to a learning activity. Cost properties are couples that may be optionally associated to learning activities, whose properties may assume values from the closed vocabulary {price, duration} and whose values are positive real numbers representing, respectively the price of a single learning resource and its average duration in minutes.

D. The Unit of Learning Model

A unit of learning represents a sequence of learning activities needed for a learner in order to understand a set of target concepts in a given domain with respect to a set of defined cost constraints [6]. It is composed by the following elements:

- a set of *target concepts* TC part of a domain model, that have to be mastered by a given learner in order to successfully accomplish the unit of learning;
- a set of *cost constraints* $CC(property) = value$ that must be taken into account in the optimisation process connected with the learning presentation generation algorithm;
- a *learning path* $LPath(c_1, \ldots, c_n)$, i.e., an ordered sequence of concepts that must be taught to a specific learner in order to let him master target concepts;
- a *learning presentation* $LPres(lo_1, \ldots, lo_m)$, i.e., an ordered sequence of learning activities that must be presented to a specific learner in order to let him/her master the target concepts.

While target concepts are defined by the course teacher, the learning path and the learning presentation are created by the generation algorithms described below. Concerning cost constraints, the property may assume values from the closed vocabulary {price, duration}. Feasible values are positive real numbers representing, respectively the maximum total price and the maximum total duration of the unit of learning.

The generation of the learning path is the starting point to completely generate a unit of learning. Starting from a set of *target concepts* TC and from a *domain model*, a feasible learning path must be generated taking into account the concepts graph $G(C, BT, IRB, SO)$ part of the domain model (with $TC \subseteq C$).

The four steps of the learning path generation algorithm are summarized below:

- The *first step* builds the graph $G'(C, BT, IRB', SO')$ by propagating ordering relations downward the hierarchical relation. IRB' and SO' are initially set to IRB and SO respectively and then modified by applying the following rule: for each arc $ab \in IRB' \cup SO'$ substitute it with arcs ac for all $c \in C$ such that there exist a path from c to b on the arcs from BT.
- The *second step* builds the graph $G''(C', R)$ where C' is the subset of C including all concepts that must be taught according to TC, i.e., C' is composed by all nodes of G' from which there is an ordered path in $BT \cup IRB'$ to concepts in TC (including target concepts themselves). R is initially set to $BT \cup IRB' \cup SO'$ but all arcs referring to concepts external to C' are removed.
- The *third step* finds a linear ordering of nodes of G'' by using a depth-first search that visits the graph nodes along a path as deep as possible. The obtained list L will constitute a first approximation of the learning path.
- The fourth step generates the final learning path $LPath$ by deleting from L all non-atomic concepts with respect to the graph G, i.e., $LPath$ will include any concept of L apart concepts b so that $ab \in BT$ for some a. This ensures that only leaf concepts will be part of $LPath$.

Successively it is necessary to generate the learning presentation suitable for a specific learner based on a *learning path* $LPath$ that has to be covered, on a set of teaching preferences TP belonging to a *domain model*, on a cognitive state CS and a set of learning preferences LP both part of the *learner model* associated to the target learner, on a set of optional *cost constraints* CC and on a set of available *learning activities*. The learning presentation is generated by following the steps below:

- The *first step* is to select the sub-list L of $LPath$ that has to be converted in a presentation. L is the sequence of all the concepts of $LPath$ not already known by the learner (i.e., any concept a so that $CS(a) < \theta$). If L is empty then the algorithm ends because the learner already knows all concepts of the learning path.
- The *second step* is to define the best sequence of learning activities P, selected from available learning activities, covering L on the basis of TP, LP and CC. This sequence definition has been given the name of *Learning Presentation Problem (LPP).*

LPP problem represents the core of this work, in fact, its solution defines the learning experiences personalization. Hereafter, LPP problem will be modeled by means of well known UFLP problem and, successively, it will be efficiently solved through an innovative memetic algorithm.

In order to model LPP by means of UFLP problem, first of all a measure of distance $d_{TP}(lo, c)$ between an activity lo and the set of preferences TP has to be defined with respect

to a concept c. In a similar way a measure of distance $d_{LP}(lo)$ based on LP may be defined. A further measure $d(lo, c)$ shall be defined as a weighted sum of the two measures.
Once the measure of distance is defined the problem of selecting the best set of activities P covering concepts of L becomes a UFLP that may be outlined with the bi-partite graph in Fig. 1, where available activities are displayed on the left and concepts to be covered on the right.

There is an arrow between a learning object LO_i (considering the available activities related to learning objects) and a subject C_j only if the metadata instance of LO_i includes a semantic link to the subject C_j

For each couple (LO_i, C_j) linked in the bipartite graph there is an assigned value $d(i, j)$ representing the distance between LO_i and C_j. Short distances define good coverings. Furthermore, to each learning object LO_i, a value $p(i)$ representing the cost of the introduction of the learning object LO_i into the sequence of LOs delivered to the learner is associated . Assume that distances $d(i, j)$ are calculated applying a specific function that evaluates the matching between the metadata values of LO_i covering C_j and the learning preferences of the learner who requests the personalized e-learning experience. Then, P must be built as the smallest set of activities covering all concepts of L with the minimum sum of distances between activities and covered concepts. Now, consider m as the number of learning objects available and n the number of subjects in the Learning Path to be filled. Set y as a binary vector where each element y_i, $(i = 1, \ldots, m)$, assumes value 1 if you decide to use the learning object LO_i, 0 otherwise, and x as binary vector in which each entry x_{ij}, $(i = 1, \ldots, m$ and $j = 1, \ldots, n)$, assumes value 1 if subject C_j is covered by the learning object LO_i, 0 otherwise. Then, the linear programming model which formalizes the whole problem is described as follows:

$$\min \sum_{i=1}^{m} p(i) y_i + \sum_{i=1}^{m} \sum_{j=1}^{n} d(i, j) x_{ij} \tag{1}$$

subject to constraints

$$\begin{array}{lll} \sum_{j=1}^{n} x_{ij} = 1 & i = 1, \ldots, m & \\ x_{ij} \leq y_i & i = 1, \ldots, m & j = 1, \ldots, n \\ x_{ij} \in \{0, 1\} & i = 1, \ldots, m & j = 1, \ldots, n \\ y_i \in \{0, 1\} & i = 1, \ldots, m & \end{array} \tag{2}$$

The optimal solution of the LPP means the identification of the optimal set of learning objects that better match (minimal distance) the learner preferences.

III. A Novel Memetic Approach to Solve Learning Presentation Problem

As any other optimization problem, LPP can be solved by means of a deterministic algorithm. This approach is valid when we consider a repository with a limited number of learning activities and subjects. However in e-Learning 2.0 scenario, the number of learning activities and subjects tends to be very high, and thus we need to look for more flexible

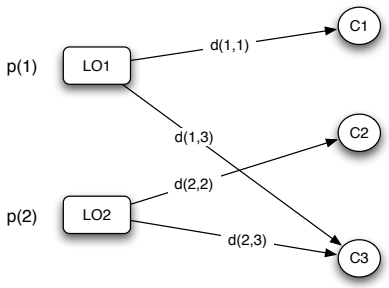

Fig. 1. Formalization of LPP.

approaches. Metaheuristics have been used extensively in the literature obtaining satisfactory results for all kind of problems. Among metaheuristics, Memetic Algorithms (MAs) [8] have become one of the most used approaches due to their ability to combine a high exploration of the search space and a high exploitation of the most promising regions.

Starting from this consideration, our proposal is to design an innovative memetic algorithm capable of efficiently generating learning presentations. In detail, this work introduces a novel parallel cooperative adaptive memetic algorithm aimed to personalize e-learning environment by combining and configuring cooperative evolutionary and local search metaheuristics in order to improve overall performance. The cooperation among the different optimization 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. With this knowledge the fuzzy engine can predict the behavior of each metaheuristic and adapt it to the presentation being personalized.

Hereafter, this section precisely describes the proposed novel multi-agent memetic algorithm capable of efficiently solving the LPP problem and, consequently, supporting e-learning environments to define high-quality personalized learning experiences.

A. A Novel Multi-agent Based Memetic Algorithm

In this section the proposed Multi-Agent Memetic Algorithm (MAMA) is introduced. MAMA is computed in two steps (see fig. 2), 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 [9] that ranks computational

performances related to evolutionary and local metaheuristics applied to a given optimization problem.

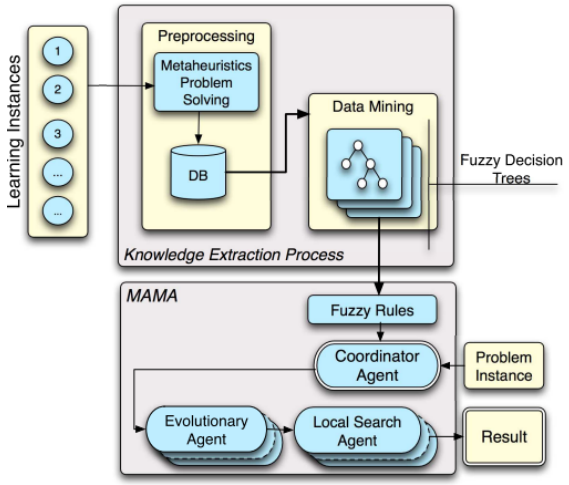

Fig. 2. Adaptive Memetic Architecture.

The parallel and distributed nature of our approach is fully suitable to be modeled using a multi-agent system [7] 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. The coordinator agent 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 an adaptive way.

The adaptability task of coordinator agent is performed by considering knowledge coming from machine learning (see section III-B) coded by means of a collection of fuzzy decision trees. In particular, trees' analysis supports coordinator agent to 1) choose the best metaheuristic parameters and 2) rank the metaheuristics suitability to solve the instance using numerical weights in the range $[0, 1]$. Through weights analysis, coordinator 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. This process allows poor metaheuristics to restart their optimization task in a more interesting point of the search space.

The cooperation is performed in the following way, optimization agents evaluate, during a certain period, 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 coordinator and optimization agents, a blackboard architecture [10] 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 the solution it has found. The coordinator *reads* this information and directs the search of each agent.

1) Modeling intelligence of the coordinator agent: A crucial question of proposed memetic architecture is how to model the coordinator agent's intelligence in order to allow it to evaluate metaheuristics performances and control the cooperation among related optimization agents. This goal is achieved by using a set of fuzzy rules able to exploit the knowledge obtained by machine learning. More precisely, the coordinator agent 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 reinitiate their search using the solution of another one which is showing a better behavior.

Coordinator agent implements first behavior by using a collection of fuzzy decision trees, as many as metaheuristics, obtained through machine learning. These trees analyze the instance in order to infer the most suitable parameters values for each metaheuristic. Moreover, the coordinator agent realizes the remaining two behaviors by considering two additional trees coming from machine learning. In detail, these trees analyze the instance and respectively rank evolutionary metaheuristics and local search metaheuristics, by returning a weights collection Ω where each $\omega_i \in \Omega$ takes a value in the range $[0, 1]$, the sum of the weights of evolutionary metaheuristics is 1, as the sum of the weights of local search metaheuristics. In short, each weight ω_i is associated with a metaheuristic and represents its suitability to solve the instance under consideration , i.e., the higher the weight of a metaheuristic the more suited it is.

In each phase, the coordinator agent uses a collection of fuzzy rules (one for each metaheuristic) exploiting Ω in order to derive the collection of poor strategies. For the sake of simplicity the rule shown below is related to evolutionary strategies (however rules acting in local phase are equivalent):

> IF [$(\omega_1 \cdot d_1$ OR ... OR $\omega_{|E|} \cdot d_{|E|})$ IS *enough*] THEN change the current solution of m_h.

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 ξ is a measure of performance shown by metaheuristics. It is 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 $[0, 1]$.

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. It should be noted that this rule tries to solve the problem that appears in parallel strategies where the exchange of solutions is unrestricted [11]

and which tend to prematurely converge to local optimums characterized by a low quality.

To perform the firing of the fuzzy rules, $\alpha - cuts$ are used. In other words, only those rules whose activation value exceeds the defined $\alpha - cut$, are fired. In the event that more than one rule is activated then the coordinator applies all of them. In this way, the coordinator agent is capable of simultaneously adapt the behavior of several poor metaheuristics.

The metaheuristics' solution exchange is performed 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 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 first phase (evolutionary phase), the poor metaheuristics' solutions changing 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 ω_k.
2) $|S| > 1$. A proportion of the worst individuals of m_h is replaced by a set of solutions where each meta-heuristic chooses its solutions as said before. The proportion is equals to $\sum_{i=1}^{|S|} \omega_i$.

Differently from first phase, during the local optimization phase the changing 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 the problem. This operation is performed in order to introduce some diversification, since the exchange of identical solutions can cause that all metaheuristics perform the search in the same space.

B. MAMA Configuration through Machine Learning

As previously mentioned the adaptability of the proposed MA is provided by the knowledge obtained through an extraction process introduced in this section. The aim of the knowledge extraction process is to evaluate the performances of metaheuristics composing the MA when applied to instances of the problem under consideration. This process returns a set of trees containing: two trees respectively modeling the suitability of the evolutionary and local search strategies by providing the aforementioned weights Ω and, on the other hand, one tree for each metaheuristic which provides the most suitable parameters choice for each metaheuristic.

The Knowledge extraction process is subdivided in two subphases, as can be seen in figure 2, Preprocessing and Data Mining.

1) Preprocessing: In preprocessing phase, metaheuristics that compose the system are applied to a set of training instances in order to build a collection of databases, named *refined databases*, that will be exploited by data mining to compute the trees that guide the coordinator. The obtaining of these databases is carried out in two steps. First, information about the performance of metaheuristics is stored in a set of so called *raw databases*, where each raw database is associated to a different metaheuristic. Successively, data contained in raw databases are processed in order to compute the final refined databases, which contain additional information useful to extract significant knowledge during data mining.

In particular, the obtaining of raw databases is carried out in the following way. For each metaheuristic and each parameter configuring this metaheuristic, a set of parameter values is chosen, after that each metaheuristic is executed k times with each possible combination of parameter values, being k an *iterative factor*, i.e., a predefined value used to obtain more accurate performance estimations. For each one of these executions a row is inserted into the raw database related to the metaheuristic being evaluated. In details, a raw database row contains:

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

Next step is to analyze raw databases in order to compute the refined databases. Refined databases are smaller than raw databases and contain additional information useful to extract significant knowledge during data mining. In detail, for each metaheuristic a refined database is computed which contains information about its best parameters combinations, i.e., which parameters values obtained a better performance. Besides, two more refined databases are computed which respectively contain information about the weights Ω associated to evolutionary and local search strategies.

In order to build databases concerning parameters for each metaheuristic its raw database is processed as follows:

1) for each instance and for each parameters combination calculate the average fitness value of its k executions.
2) for each instance select the parameter combination computing the best average fitness value;
3) the refined database is updated with a new entry containing the description of instance and the best parameter combination.

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

1) for each metaheuristic, for each instance and for each parameters combination calculate the average fitness value of its k executions.
2) for each metaheuristic and each instance, select the best average fitness value and compute the *metaheuristic suitability weight* ω_i as:

$$\omega_i = \frac{fit^i_{best}}{\sum_{l=1}^{\#evolutionary\_strategies} fit^l_{best}}$$

3) for each metaheuristic and each instance the refined database is updated with a new entry containing the de-

scription of the instance and the metaheuristic suitability weight ω_i.

Database associated to local search methods is analogously built by replacing evolutionary strategies with local search methods. Refined databases represent the output result of Preprocessing phase and they will be used by a data mining technique to complete the Knowledge Extraction Process in computational efficient way.

2) Data Mining: Once gathered performance information and collected it through refined databases, the Data Mining phase extracts the collection of knowledge models through data mining. Our data mining approach exploits a fuzzy decision tree (FDT) technique [12] to extract knowledge from refined databases and build the aforementioned collection of trees. Two main reasons have guided the choice of FDTs. First their interpretability, decision trees stand out on account of their simplicity and readability, which is an important issue in data mining, and enable a full understanding of the knowledge that guides the coordinator. Second, their fuzziness, i.e., the possibility of dealing with fuzzy theory, which can improve inference performance by means of approximate reasoning and provide an easy way to obtain the weights Ω, one of the most important parts of coordinator's knowledge.

Once this phase is finished the collection of fuzzy decision trees that models coordinators's knowledge, is obtained, and it is ready to control the execution of the optimization agents.

IV. Experimental Results

In this section we want to asses the validity of the proposed approach, MAMA. The main purpose of MAMA is to help improving the e-learning process. For this reason we plan to include it as a plug-in of a learning management system. In particular, we plan to include it in Sakai project[1]. Sakai is an open source learning management which enables powerful teaching, learning and research collaboration within a feature-rich, enterprise platform. However, Sakai does not provide the possibility of automatically personalize e-learning experiences in the previously explained way. We consider this an essential feature for modern e-learning solutions, and thus believe that the development of a plug-in that adds this feature can improve learners' experience. In the design of the planned plug-in, a crucial issue has been 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). Other metaheuristics could have been chosen or even added, but as a first step we have chosen these four.

Hereafter, the results of some preliminary performance tests and a study with real learners will indicate the positive impact of MAMA in the process of e-learning.

[1]www.sakaiproject.org

TABLE II
Parameters values for each metaheuristic

GA		*PSO*	
parameter	*value*	*parameter*	*value*
Selection op.	Roulette	Topology	Local Best 2N
Cross prob.	0.6	ω	0.729844
Mutation prob.	0.01	r_p	2
Population size	100	r_g	2
		# of particles	100
TS		*SA*	
parameter	*value*	*parameter*	*value*
Tabu list size	10	Cooling schedule	Hyperbolic Tangent
Diversification t.	50	Inner iterations	5000

A. Computational experiments

In this subsection, we perform some tests in order to assess the validity of MAMA, comparing its results with those obtained by the different non-adaptive MAs that can be obtained from the combination of the single metaheuristics that compose MAMA in a sequential way, 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).

The parameters used by each metaheuristic are shown in Table II.

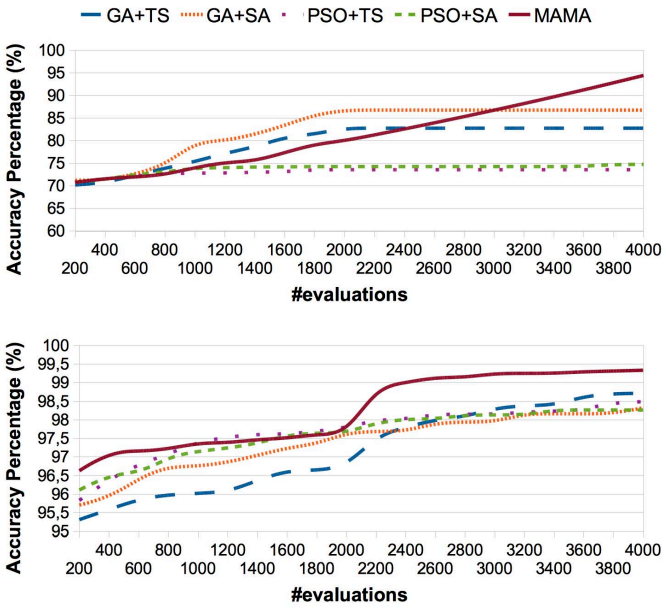

Fig. 3. Evolutions of the different strategies.

1) Performance tests: As a first approach it would be interesting to perform some preliminary tests in order to asses the validity of the proposed strategy. In order to perform these tests we have created an instance generator able to construct artificial instances of the learning presentation problem. We

have identified some instance's features that may influence the behavior and performance of different solving strategies:

- Number of learners.
- Ratio between learning objects and learners.
- Connection percentage between learning objects and learners.

We have carried out two tests with two different instances generated using the aforementioned generator. The purpose of these tests was to graphically interpret the behavior of the different strategies during their search of the optimum. The results are shown in Fig. 3, this figure illustrates the evolution of the 5 strategies. The graphics plot the fitness value obtained by each strategy at each moment of the execution, showing the average of 10 executions of 4000 evaluations. From these graphics we can extract interesting information about the behavior of MAMA. In particular, it can be observed that although MAMA's convergence is not the fastest, it obtains in both instances the best performance. These are promising results which seem to indicate that MAMA solves e-learning LPP problem in a more efficient way than other static memetic approaches and, as will be shown afterwards, it generates very suitable personalized learning experiences.

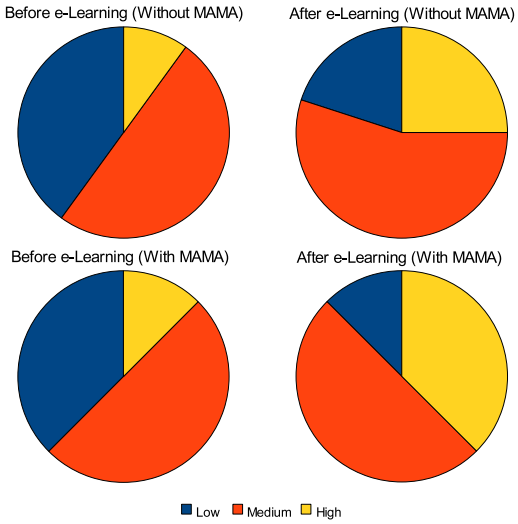

Fig. 4. Experimentation results on a group of 38 learners with and without the proposed memetic Sakai plug-in prototype.

2) Learning Comparative Studies: Once we have demonstrated the effectiveness of MAMA with artificial instances, we perform a real world test, involving real learners that want to train on enterprise management. The control and experimental groups were made up of 20 and 18 learners respectively. Data corresponding to the control group comes from historical data from a previous experiment, and shows the results of learning using Sakai without the help of MAMA. On the other hand, data corresponding to experimental group was obtained from the results achieved by a new group of learners, which were able to use the capabilities of MAMA. All the voluntary learners were tested before and after a training phase on the same topics. In all the tests the learners' skills in the chosen domain were quantified using three ability ranges: low-level (0-3 scores), medium-level (4-7 scores) and high-level (8-10 scores). Figure 4 presents a summary of the results. As can be observed, both e-Learning approaches provide the necessary tools to improve learners' knowledge, in fact the quantifications of the knowledge are statistically better with a significance of the 99.9% after using e-learning. Moreover those learners that have used MAMA have obtained better results than those which not, with an statistical significance of the 95%. These results are very promising and support the creation of a plug-in for Sakai able to include MAMA's capabilities.

V. Conclusions

Recent advances in Web 2.0 have impulsed the development of the so called e-learning 2.0. In this context, Learning Management Systems should promote new e-learning trends by means of suitable models and processes that dynamically and intelligently create personalized e-learning experiences adapted to learner expectations and objectives. Our work proposed an innovative memetic algorithm capable of efficiently generating personalized learning presentations. In detail, we have proposed a parallel cooperative adaptive memetic algorithm aimed to personalize e-learning environment by combining and configuring cooperative evolutionary and local search metaheuristics in order to improve overall performance. The cooperation 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. As shown by experimental results, our proposal enhances significantly the learning framework in terms of flexibility and efficiency.

References

[1] M. Ebner, "E-Learning 2.0 = e-Learning 1.0 + Web 2.0?," in *Proc. 2nd Int. Conf. Availability, Reliability and Security*, pp. 1235–1239, 2007.

[2] T. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," *Int. J. Human-Computer Studies* vol. 43, pp. 907-928, 1994.

[3] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacited facility location problem," *Discrete Location Theory*, P. Mirchandani and R. Francis (Eds.), John Wiley and Sons Inc., New York, pp. 119-171, 1990.

[4] G. Albano, M. Gaeta, and S. Salerno, "E-learning: a model and process proposal," *Int. J. Knowledge and Learning*, vol. 2, pp. 73–88, 2006.

[5] G. Albano, M. Gaeta, and P. Ritrovato, "Testing hypotheses in the functional linear model," *Scandinavian J. Statistics*, vol. 30, pp. 241–251, 2007.

[6] G. Acampora, M. Gaeta, V. Loia, P. Ritrovato, and S.Salerno, "Optimizing Learning Path Selection through Memetic Algorithms," *IEEE Int. Joint Conf. Neural Networks*, pp. 3869–3875, 2008.

[7] M.J. Wooldridge, *Multi-agent Systems : An Introduction*, John Wiley and Sons, 2002.

[8] 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.

[9] J.M. Cadenas, M.C. Garrido, and E. Muñoz, "Using machine learning in a cooperative hybrid parallel strategy of metaheuristics," *Information Sciences*, vol.179, pp. 3255–3267. 2009.

[10] Daniel D. Corkill. "Blackboard Systems," *AI Expert*, vol. 6, no. 9, pp. 40-47, September 1991.

[11] T. G. Crainic, M. Gendreau, P. Hansen, and N. Mladenovic, "Cooperative parallel variable neighborhood search for the p-median," J. Heuristics, vol. 10, pp. 293–314, 2004.

[12] C.Z. Janikow, "Fuzzy decision trees: issues and methods," *IEEE Trans. Syst. Man Cybern. B, Cybern.*, vol. 28, no. 1, pp. 1–14, 1998.