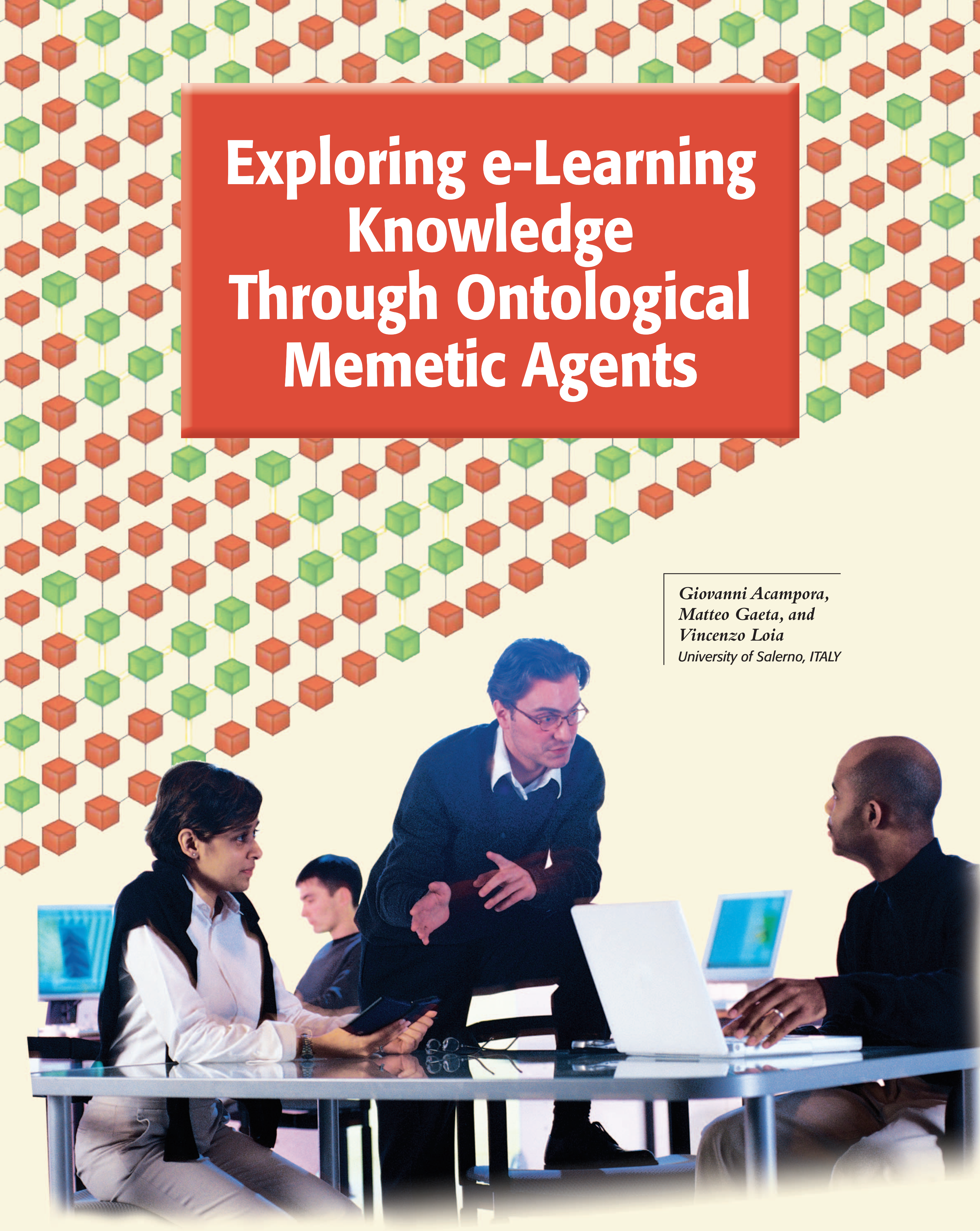
Exploring e-Learning Knowledge Through Ontological Memetic Agents

Giovanni Acampora, Matteo Gaeta, and Vincenzo Loia
University of Salerno, ITALY



Digital Object Identifier 10.1109/MCI.2010.936306

1556-603X/10/$26.00©2010IEEE

Abstract: E-Learning systems have proven to be fundamental in several areas of tertiary education and in business companies. There are many significant advantages for people who learn online such as convenience, portability, flexibility and costs. However, the remarkable velocity and volatility of modern knowledge due to the exponential growth of the World Wide Web, requires novel learning methods that offer additional features such as information structuring, efficiency, task relevance and personalization. This paper proposes a novel multi-agent e-Learning system empowered with (ontological) knowledge representation and memetic computing to efficiently manage complex and unstructured information that characterize e-Learning. In particular, differing from other similar approaches, our proposal uses 1) ontologies to provide a suitable method for modeling knowledge about learning content and activities, and 2) memetic agents as intelligent explorers in order to create "in time" and personalized e-Learning experiences that satisfy learners' specific preferences. The proposed method has been tested by realizing a multi-agent software plug-in for an industrial e-Learning platform with experimentations to validate our memetic proposal in terms of flexibility, efficiency and interoperability.

I. Introduction

In the last years the impulse carried out by the so-called Web 2.0 [1] has involved also the e-learning field where new trends are rising. Tools like Blogs (used to share ideas), Wikis (used as a way to construct knowledge in a collaborative way), Podcast (used to distribute multimedia files over the Internet) and other Web Sharing Applications (e.g. Flickr, YouTube, del.icio.us, etc.) allow Internet communities to work, teach, learn, conduct business, etc. The coherent utilization of the aforementioned tools in e-learning processes is called e-learning 2.0 [2]. In this vision, the distributed nature of the Web brings several criticisms like the management of users' identities across different Web Applications, the efficient management of several content repositories, the harmonization of contents coming from heterogeneous Web sites and so on.

In order to overcome the well recognized limitations of the learning paradigm supported by the current commercial Learning Management Systems (LMS) [3] and to provide new e-learning trends, we need suitable models and processes that dynamically and intelligently structure distributed e-learning knowledge and create e-learning experiences (i.e. a structured collection of content and services able to facilitate learners in acquiring a set of competencies about a specific domain) adapted to learner expectations and objectives in the new Web environment.

Ontologies and memetic agents are a suitable integrated approach for defining personalized e-learning experiences, i.e., most fitting sequences of learning activities able to maximize the understanding level of learners with respect to specific learning objectives. Indeed, ontologies can model and represent the educational domains of interest realizing their conceptualization by identifying its relevant subjects and organizing them by means of a fixed set of relations [4] as depicted in many fields of knowledge engineering [5][6][7]. At the same time, memetic agents can be considered as knowledge explorers capable of analyzing ontologically structured knowledge and inferring additional information that improve learner's understanding capabilities.

In this work, memetic agents' exploration of taxonomic knowledge is formalized as an optimization problem. Indeed, a possible approach to face memetic exploration in ontological e-Learning context is to consider and solve the well-known *Plant Location Problem (PLP)* [8]. In particular, we propose the use of a novel distributed *memetic algorithm*, based on a multi-island idea, capable of efficiently solving the e-learning PLP problem and computing good quality personalized learning experiences. Different from previously proposed multi-island systems, our memetic agents explore learning knowledge in an optimal way by taking into account hardware details of islands composing the framework. This enables a massive parallelism that speeds up the generation of personalized e-learning presentations. Experimental results show that memetic exploration can find suitable sub-optimal solutions with a smaller computational effort than the classical optimization approach. Consequently, the proposed approach is particularly convenient when applied in Web 2.0 scenarios where distributed repositories and the learning paths are made by numerous subjects. Metaphorically speaking, our approach tries to achieve aforementioned results by building a so-called e-Learning Mesh, a set of knowledge highways whose paths connect information sources and learner's requirements and cross feasible learning contents; at the same time, memetic agents are computational entities that parallel pass through highways' paths and compute high-quality learning roads.

This paper is organized in four sections. In Section II an overall view of our personalized e-learning approach is presented; in Section III, an optimization problem characterizing the generation of near best e-learning experience binding is

E-Learning systems have proven to be fundamental in several areas of tertiary education and in business companies.

formalized; in Section IV, a computationally efficient, distributed memetic solution for the e-learning experience binding problem is furnished; the Section V reports the experimental results obtained from proposed optimization approach.

II. Memetic Algorithms and Multi-Agent Systems: A Novel Distributed Approach of Optimization

Multiagent Systems (MAS) is the subfield of Artificial Intelligence whose main aim is to provide both principles for construction of complex systems involving distributed multiple agents and mechanisms for coordination of independent agents' behaviors [9]. Though there is no generally accepted definition of "agent" in AI [10], for the our purposes, we consider an agent to be an entity, such as a robot, with goals, actions, and domain knowledge, situated in an environment. The way it acts is called "agent's behavior" and it is characterized by several important characteristics [11]:

❑ Autonomy: the agents are at least partially autonomous;
❑ Local views: no agent has a full global view of the system, or the system is too complex for an agent to make practical use of such knowledge;
❑ Decentralization: there is no designated controlling agent (or the system is effectively reduced to a monolithic system) [12].

Through a composed agent's behaviors, an MAS is capable of providing different design benefits such as parallelism, robustness, scalability, geographic distribution and cost effectiveness [13] and, consequently, it is used in several application domains as, for example, the wholesale power market performance measures [14]. The advantages provided by an MAS became more evident in 2005 when Intel Corporation followed the lead of IBM's Power 4 and Sun Microsystems' Niagara processor in announcing that its high performance microprocessors would henceforth rely on multiple processors or cores. The new buzzword *"multicore"* highlights the plan of doubling the number of standard cores per die with every semiconductor process generation starting with a single processor [15]. In this novel scenario of distributed computing, an MAS amplifies their benefits by taking into account systems' hardware details in order to parallel complex tasks by exploiting opportune organizational paradigms. In particular, multicore processors allows MAS to organize distributed computing in a hierarchical way where agents are conceptually arranged in a treelike structure where each node represents a processor core. In this hierarchy, agents higher in the tree have a more global view than those below them. The data produced by lower-level agents in a hierarchy typically travels upwards to provide a broader view, while control flows downward as the higher level agents provide direction to those below. As will be shown in this paper, this novel scenario of Hierarchical Multicore MAS is well suitable to model optimization schemes based on memetic theory.

The term *meme* was coined by Richard Dawkins to define "the basic unit of cultural transmission or imitation" [16]. In the computational intelligence context, the new area of memetics computing is a novel class of hybrid evolutionary optimization algorithms based on the exploitation of methods such as the local improvement procedure [17][18].

In our present scenario, the Memetic Computing methodology materializes in the form of hybrid global-local heuristic search. In particular, the global search is a form of population based method, while the local search or meme, is a typical local search operator as tabu search, simulated annealing or similar.

The hybridization scheme is particularly apt to be embedded into a hierarchical multicore MAS where higher-level agents split candidate population into a collection of subpopulations that are sent downward in the hierarchy [19]. Once lower-level agents receive their population portion, they compute some form of population-based optimization and send the computed results towards higher levels of the hierarchy where agents act as memes and apply local search operators on the incoming data. The coordination and cooperation among optimization agents enable a fine exploration of the problem population and thus achieving a considerable convergence speed-up and, at the same time, arriving at high quality solutions.

In this paper, a hierarchical MAS is introduced with a collection of cooperating memetic agents as a suitable integrated approach for exploring e-Learning information and defining personalized e-learning experiences that represent the most suitable sequences of learning activities able to maximize the understanding level of learners with respect to specific learning objectives.

III. E-Learning Mesh and Knowledge Highways: An Ontological Educational Environment

In order to explore e-Learning information and generate personalized experiences, it is first of all, necessary to formalize the learning knowledge. Indeed, as shown in Fig. 1, the multitude of *information sources* proposed by Web 2.0 (Google, Youtube, Wikipedia, Bing and so on) leads to new learning contexts characterized by a plethora of unrelated information that are not able to support learners. The formalization task can be performed through a collection of well-defined models structured by exploiting standard technologies for systematic representation of knowledge: ontologies. Our goal is to describe a methodology to organize this information by means of a collection of e-Learning models whose relationships define the so-called *e-Learning Mesh*.

E-Learning Mesh can be view as a set of highways whose paths connect different pairs of cities and cross several intermediate points. From the learning point of view, the ends of paths are, respectively, information sources and learner's requirements; intermediate points are learning contents satisfying requirements by using those sources; highways can be regarded as routes useful to improve learners' skill and competence in intelligent and short time way. However, in order to achieve this result the most suitable sequence of intermediate points have to be chosen and consequently, an efficient method capable of examining the mesh and returning those points is needed. As will be shown in the following, memetic agents represent an appropriate method to explore the mesh and derive sub-optimal paths from mesh. Metaphorically speaking, these agents cooperate in order to simultaneously explore several portions of highway network and build different sub-paths. Successively, agents communicate to connect these sub-paths and define the final solution. Fig. 2 shows a simple example of a bi-dimensional e-Learning Mesh together with a sample of a high-quality e-learning experience taking into account the information sources and learner's preferences.

Next section is devoted to introduce the *E-Learning Models*, ontological templates useful to build cities belonging to highway paths and to define the *learning presentation generation algorithm*, including a formal representation of metaphoric highway exploration performed by memetic agents.

A. E-Learning Models

Proposed e-Learning system is based on a set of models that is able to represent the main entities involved in the process of teaching/learning and on a set of methodologies, leveraging on such models, for the generation of individualized learning experiences with respect to learning objectives, learners' knowledge and learning preferences. These models can be viewed as templates useful to define cities belonging to previously mentioned highway paths together with roads which connect them, i.e. learning experiences. Our solution adopts four models [20]:

❑ the *domain model* represents the knowledge domain that is the 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 [21];
❑ 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.

Our system uses these models to automate some of the phases of the teaching/learning process. In particular the teacher may initialize a unit of learning by setting target concepts and associate one or more learners to it (in self-directed learning, learners settle own target concepts and constraints by themselves). Then the system generates a personalized *learning path* (sequence of concepts to be thought) 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 generates the best *learning presentation* (sequence of learning activities) for each enrolled learner by applying the *learning presentation generation algorithm*. The learner then undertakes the learning and testing activities of the learning presentation until completion. 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).

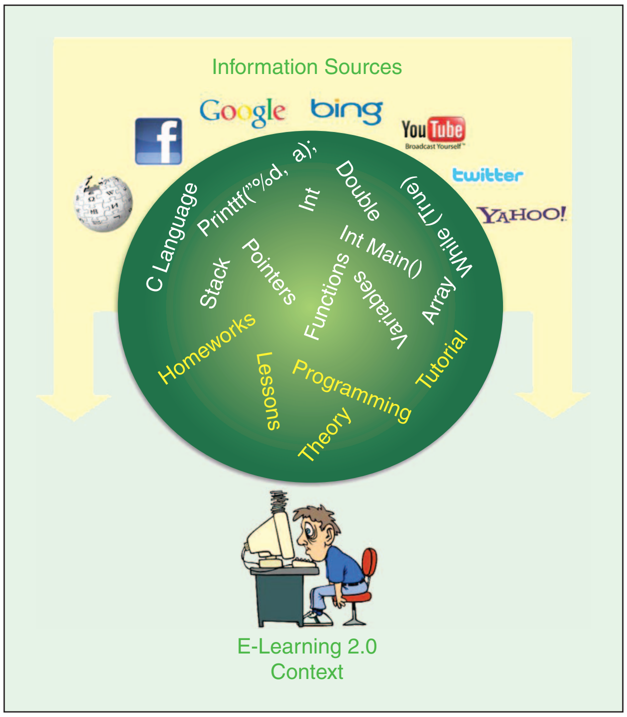


FIGURE 1 E-Learning 2.0 scenario: several sources can provide information about "C language" and learner can assimilate it in different way. No order is defined.

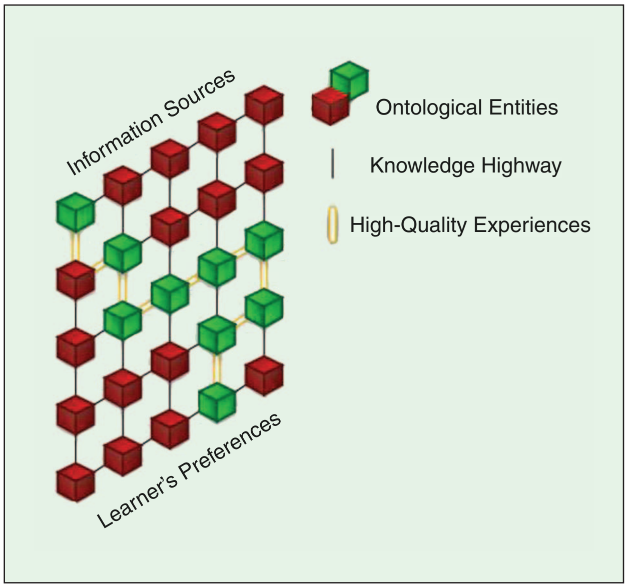


FIGURE 2 An example of bi-dimensional e-Learning Mesh.

In the following paragraphs we outline the main models and algorithms applied by the system (for the sake of conciseness we omit some details like test management and learner model evaluation and update). In the last paragraph we then introduce some problems concerning the learning path generation algorithm that are originally solved in the remaining part of the paper.

1. 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. In other words, it describes in a formal way, information coming from the aforementioned Web 2.0 sources. The mechanism used is named ontology, i.e. an engineering artefact, constituted by a specific vocabulary [22] used to describe a certain reality, plus a set of explicit assumptions regarding the intended meaning of the vocabulary words [23].

In our approach the ontology is composed by a set of concepts (representing the topics to be taught) and a set of relations between concepts (representing connections among topics). Such 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 kind 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 is understood if each a so that a belongs to b is understood (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 favor learning, it is desirable to study a before b (ordering relation).

Fig. 3 shows a sample domain model in the didactics of artificial intelligence exploiting the relations defined above and stating that to understand "logics" means to understand "formal systems", "propositional logic" and "first order logic" but, before approaching any of these topics, it is necessary to have an "outline of set theory" first. Moreover, "formal systems" must be taught before both "propositional logics" and "first order logic" while it is desirable (but not compulsory) to teach "propositional logics" before "first order logic".

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 an application $TP(C \times Props \times PropVals) \rightarrow [0, 10]$ where *Props* is the set of didactical properties and *PropVals* is the set of feasible values for such properties. Table 1 provides some (non-exhaustive) examples 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 1.

2. 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 [24]. The *cognitive state* represents the knowledge reached by a learner at a given time and it is represented as an application $CS(C) \rightarrow [0, 10]$ 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, a value of 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 1 for teaching preferences, whereas the values 0 and 10, represent, respectively, the lowest and highest level of learning preference. Different 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 transforming them in suitable values for learning preferences. Both parts of the learner model are automatically updated by the system during learning activities.

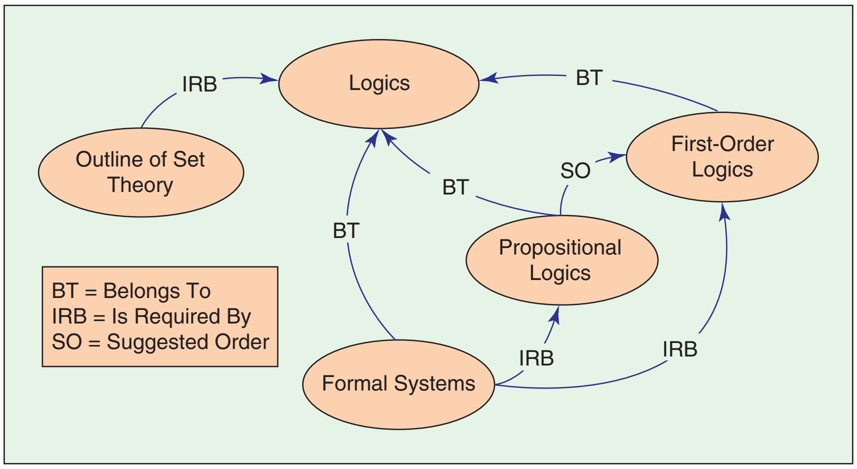


FIGURE 3 A sample domain model.

3. The Learning Activities 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, video clips, podcasts, simulations, exercises, etc.) or learning services (e.g. virtual labs, wikis, folksonomies, forums, etc.). Our system uses learning activities as building blocks 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} part of a given domain model, that is covered by 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 optimization 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 1). Different 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 with 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.

4. 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 [25], [26]. It is composed by the following elements:

- a set of *target concepts* TC part of a domain model, that has 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 optimization 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.

TABLE 1 Example of didactical properties and feasible values.

PROPERTIES	FEASIBLE VALUES
DIDACTIC METHOD	DEDUCTIVE, INDUCTIVE, ETC.
ACTIVITY TYPE	DISCUSSION WITH A PEER, DISCUSSION WITH THE TEACHER, ETC.
INTERACTIVITY LEVEL	HIGH, MEDIUM, LOW

5. The Learning Path Generation Algorithm

Having described the basic elements useful to define the main components of e-learning knowledge (see Fig. 4), it is necessary to specify meaningful relationships joining them: learning paths. The generation of the learning path is the first step 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 exists a path from c to b on the arcs from BT.

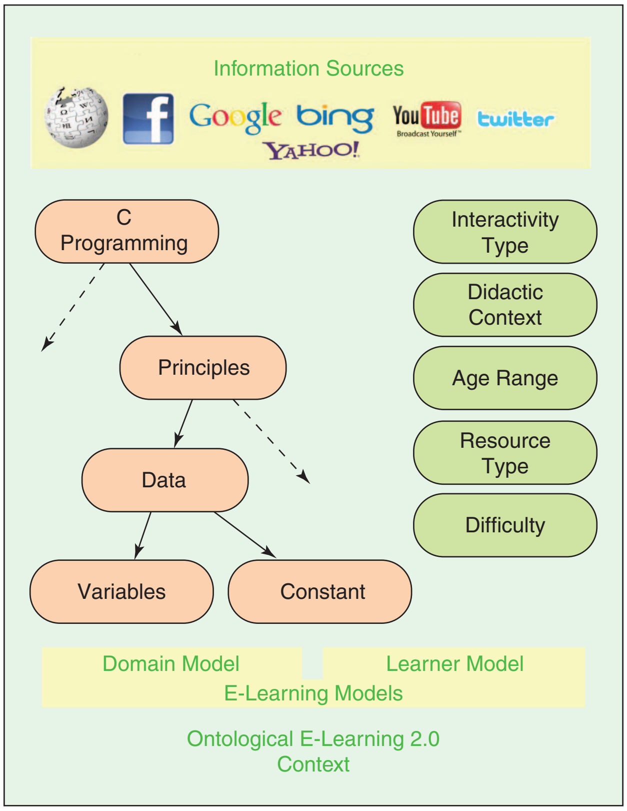


FIGURE 4 The e-Learning 2.0 scenario with e-Learning models: C language" and learner ontology support student to can assimilate information.

- ❑ 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 a 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 the depth-first search algorithm. 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*.

As an example we may consider the concept graph in Fig. 3 as G and {*"Logics"*} as TC. Following the steps above, to understand logics, the learner has to learn the outline of set theory, then formal systems, then propositional logics and, finally, first order logics so *LPath* = ("*Outline of Set Theory*", "*Formal Systems*", "*Propositional Logics*", "*First Order Logics*"). If we consider {First Order Logics} as TC, instead, the algorithm result is: *LPath* = ("*Outline of Set Theory*", "*FormalSystems*", "*First Order Logics*") meaning that propositional logic is not a strict requirement for first order logic.

6. The Learning Presentation Generation Algorithm

The joint usage of e-learning models and learning paths defines the aforementioned E-Learning Mesh, the collection of knowledge highways whose exploration is defined by the presentation generation algorithm. The presentation generation algorithm is purposed to build a presentation that is part of a unit of learning that is suitable for a specific learner. The presentation generation algorithm computes a learning presentation by exploiting several factors such as:

1) a *learning path LPath* that has to be covered,
2) a set of teaching preferences TP belonging to a *domain model*,
3) a cognitive state CS and a set of learning preferences LP, both part of the *learner model* associated to the target learner,
4) a set of optional *cost constraints* CC on a set of available *learning activities*.

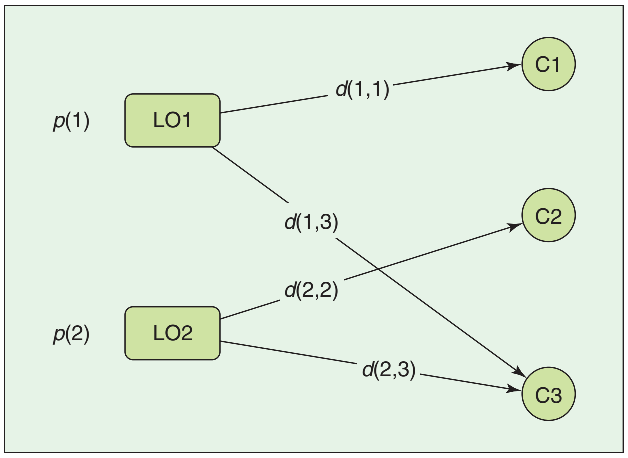


FIGURE 5 Formalization of a plant location problem in e-learning context.

The steps of the presentation generation algorithm are summarized 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.

To solve the second step, 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 PLP that may be outlined with the bi-partite graph in Fig. 5 where available activities are displayed on the left and concepts to be covered on the right. 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.

In the situation depicted in Fig. 5 we have two learning objects to use in order to explain a set of three subjects. There is an arrow between a learning object LO_i 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 $d(i, j)$. The value $d(i, j)$ represents the distance between LO_i and C_j. Short distances define good coverings. Furthermore, to each learning object LO_i is associated 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. Let's 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.

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_i, $(i = 1, \ldots, m)$, a binary vector that assumes value 1 if you decide to use the learning object LO_i, 0 otherwise, and x_{ij}, $(i = 1, \ldots, m \text{ and } j = 1, \ldots, n)$, is a binary vector assumes value 1 if subject C_j is covered by the learning object LO_i, 0 otherwise. The linear programming model which formalizes the whole problem is described as follows:

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

subject to constraints

$$\begin{aligned}
&\textstyle\sum_{j=1}^{n} x_{ij} = 1 && i = 1, \dots, m \\
&x_{ij} \le y_i && i = 1, \dots, m \quad j = 1, \dots, n \\
&x_{ij} \in \{0,1\} && i = 1, \dots, m \quad j = 1, \dots, n \\
&y_i \in \{0,1\} && i = 1, \dots, m.
\end{aligned}$$

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

IV. Generating Personalized E-Learning Experience through Memetic Agents Exploration

In previous section the formalization of e-learning knowledge has been introduced. This knowledge has been exploited to realize a collection of paths, connecting learning objects and contents, modeled by means of the PLP problem. In this section a methodology based on memetic optimization is introduced in order to explore different learning paths and return the best learning presentation which maximizes learner's understanding capabilities (see Fig. 6). The methodology is based on the joint use of multi-agent systems and memetic computation. This integration results in a parallel method of learning paths exploration able to:

1) generate best personalized e-learning experiences;
2) minimize the computational effort necessary to compute the optimal matching.

The benefits obtained from this hybrid technique support, mainly, the e-learning 2.0 frameworks where the number of information sources is very large. More in detail, our approach represents an extended version of multi-island genetic algorithms [27] [28] where a collection of agents is devoted to parallel explore a problem's search space distributed in a heterogeneous computer network. The extended model that we propose introduces a novel idea of agent whose behavior is dependent upon the hardware configuration of the host computing it. Indeed, our memetic agents deal with multi-core processors in order to explore problem solutions in a more efficient way.

A. An Extended Island Model for Parallel Memetic Algorithms

Our proposal of memetic optimization exploits a heterogeneous computer network organized in a star topology (Fig. 7), composed by $n + 1$ hosts, where the first n hosts are devoted to performing our evolutionary parallel approach, whereas, the additional host, named *supervisor*, is responsible for administrative aims as the population distribution, the islands' synchronization

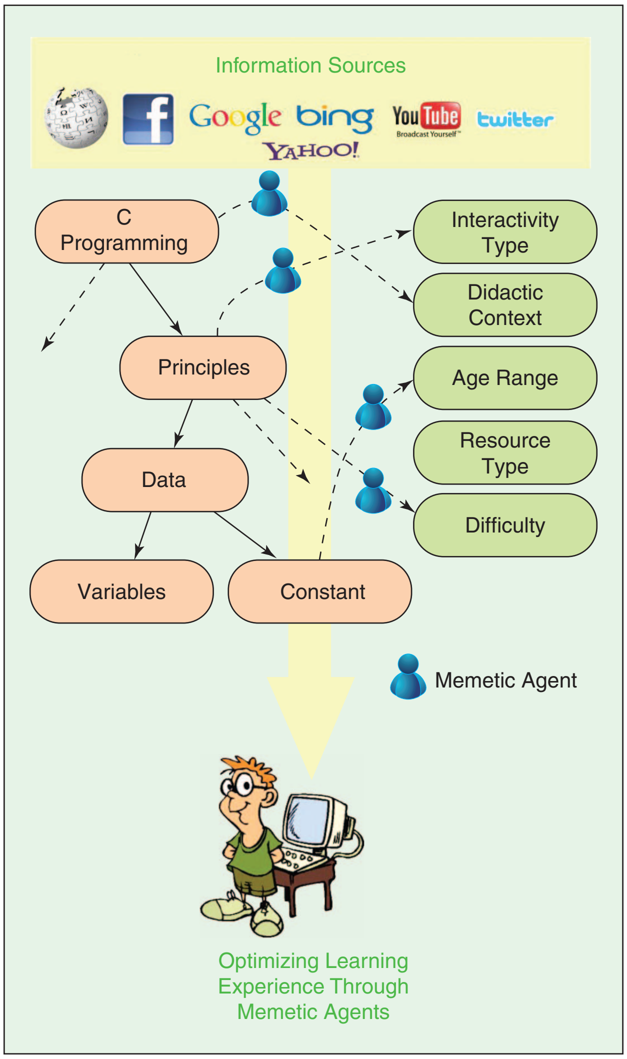


FIGURE 6 Memetic agents exploration for deriving optimized learning paths.

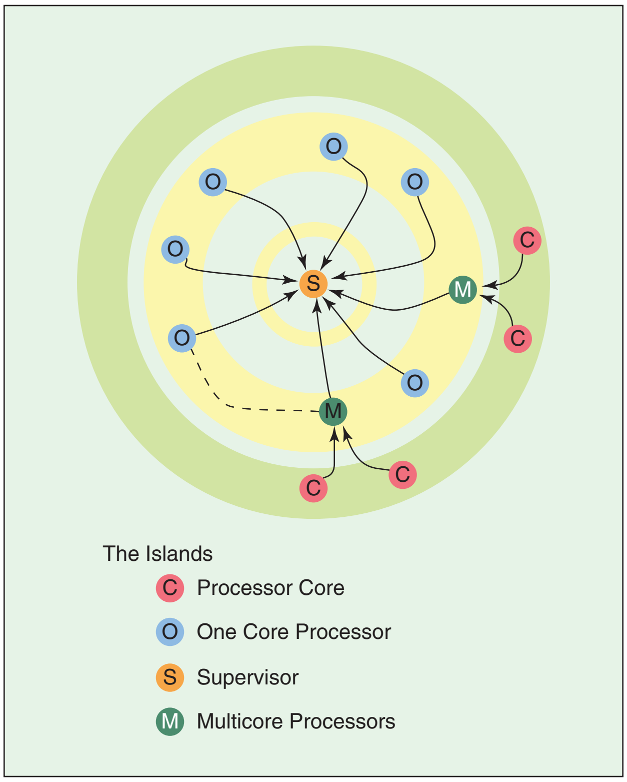


FIGURE 7 The network topology of the extended island model.

When our approach is used to model an e-Learning 2.0 scenario, the related PLP problem becomes too complex to be solved through a deterministic approach or a sequential evolutionary algorithm.

and the application of a complete local search strategy. Our method partitions the n hosts into two categories named, respectively, as I_1 and I_2 where I_1 is the collection of single core processors, whereas, I_2 is the collection of multi cores processors. The number of islands is defined as follows:

$$|\text{Islands}| = |I_1| + \left(\sum_{p_j \in I_2} \text{cores}(p_j)\right) + 1,$$

where *Islands* is the collection of computational entities composing our model and *cores* (p_j) is a function returning the number of cores of the processor p_j. The network hosts can adopt two different behavioral strategies: the *supervisor* and *genetic* behaviors.

1. Supervisor Behavior

The supervisor is an agent responsible for generating an initial collection of candidates' solutions of a given optimization problem P. Let pop be the generated population. Successively, the supervisor distributes $|\text{pop}|$ chromosomes on the $(|I_1| + |I_2|)$ hosts by following a uniform approach in order to obtain a collection of $(|I_1| + |I_2|)$ disjoint population named pop_i with $i = 1, \ldots, (|I_1| + |I_2|)$. Each host will manage with at most $\lceil |\text{pop}|/(|I_1| + |I_2|) \rceil$ candidate solutions. The host i computes the genetic behavior on its population portion and, successively, it returns the best best_i solutions selecting them by means of a Gaussian approach. Once received the $\sum_{i=1}^{|I_1|+|I_2|} \text{best}_i$, the supervisor applies additional genetic evolutions on the best individuals coming from different species (the optimized subpopulations) together with a local search strategy such as simulated annealing, tabu search or other personalized refinements in order to eventually improve the islands best solution.

2. Genetic Behavior

The hosts implementing the genetic behavior are agents capable of applying two operations in a sequential way:

❑ Hierarchical distribution among processor cores ;
❑ Genetic operators: crossover, mutation and migration.

The hierarchical distribution is performed only if it is necessary, i.e., only if the system is composed by more than one core; if pop_i is the population associated with ith host and cores_i is the number of processor cores, then the genetic behavior generates cores_i population, pop_{ij} with $j = 1, \ldots, \text{cores}_i$ and it distributes the novel populations on the processor cores. Formally,

$$\text{pop}_i = \bigcup_{j=1}^{\text{cores}_i} \text{pop}_{\text{ij}} \text{ with } i \in \{1, \ldots, |I_1| + |I_2|\}.$$

Once received the subpopulation pop_{ij}, the jth core applies the genetic operators in a standard way in order to improve the fitness mean value of pop_{ij}; let $\mu_{\text{pop}_{ij}}$ be this mean value. The island uses the mean values $\mu_{\text{pop}_{ij}}$ with $j = 1, \ldots, \text{cores}_i$ in order to choose the best chromosomes belonging to core populations. These chromosomes are chosen in a selective way by exploiting the following Gaussian distribution:

$$\gamma_i(\mu_{\text{pop}_{ij}}) = \left(\frac{1}{\sqrt{2\pi} \cdot \sigma} \exp\left(-\frac{1}{2}\left(\frac{\mu_{\text{pop}_{ij}} - \mu}{\sigma}\right)^2\right)\right) \cdot \eta,$$

where $\mu_{\text{pop}_{ij}}$ is the mean fitness value coming from jth core, μ is the mean value, σ is the standard deviation, η represents a scaling factor. Then, the number of candidate solutions to apply local search, best_{ij}, is defined by the previous Gaussian function, the mean fitness value and the size of subpopulation pop_{ij}:

$$\text{best}_{ij} = \gamma_j(\mu_{\text{pop}_{ij}})^* |\text{pop}_{ij}|.$$

Then the total number of chromosomes (rounded up) returned to the supervisor island is:

$$\text{best}_i = \sum_{j=1}^{\text{cores}_i} \lceil \text{best}_{ij} \rceil,$$

where best_{ij} denotes the number of the best candidates coming from the jth cores on the ith island. The choice of μ, σ, η has to be done with attention because this choice determines the best_{ij} value. If best_{ij} is small then the mean value is very high but the number of solutions that are proposed to improve their fitness is very small, vice versa, if best_{ij} is large then its mean value is not very high but the population contains a high number of chromosomes that are proposed to improve their quality. If the island implementing the genetic behavior is hosted on a single core processor then the hierarchical distribution is omitted and the classical sequence of genetic operators is applied and the same Gaussian function is used to select the best best_i chromosomes from pop_i.

B. Applying the Extended Island Model for Memetic Algorithm to e-Learning Scenario

As shown in previous section our e-Learning system defines a PLP optimization problem associating a collection of didactic concepts to a set of learning objects. The weight of the arcs connecting concepts and learning objects are defined by exploiting the user learning preferences, whereas, the concept costs are derived from a combination of multiple factors. When our approach is used to model an e-Learning 2.0 scenario, the related PLP problem becomes too complex to be solved through a deterministic approach or a sequential evolutionary algorithm. In this section the extended island model defined in the previous section will be applied to the e-Learning PLP problem by defining the chromosome template, the collection of genetic operators exploited by islands, the local search strategy used by the supervisor island and the set of

Gaussian distribution parameters used to derive the best matching among learning objects and didactic activities. In order to solve the PLP problem previously defined, it is necessary to fix different genetic properties and parameters involved in the evolutionary schemas. In details we have to define *i*) how to represent a potential problem solution, i.e., a chromosome, *ii*) the genetic operators to exploit in order to run genetic evolutions and *iii*) the fitness function indicating how good the solutions computed by the algorithm are. As previously stated, a solution of the defined problem is given by the optimal allocation of m didactic objects to n learning concepts where each concept has to be covered by one and only one didactic activity. In other words, a chromosome L can be defined as an integer values vector composed by n components (genes) range in $[1,m]$, where the ith gene $l_i = j$ means that jth learning concept is covered by the ith learning object (i.e. $p(i) = 1$ and $d(i,j) \geq 0$). In this way the algorithm fitness function (to minimize) is

$$\text{fitness}(L) = \sum_{i=1}^{m} p(i) + \sum_{i=1}^{m}\sum_{j=1}^{n} d(i,j)$$

with only one constraint: if c is a given chromosome and, $c[i]$ and $c[j]$ with $j > i$ are two learning objects related with concept i and j and $c[i] = c[j]$ then, necessarily, it must be $j = i + 1$. To test the solution's feasibility, $O(n^2)$ computational steps are necessary, where n is the chromosome size. In fact, the constraint can be checked only by analyzing each of n^2 gene pairs. If we suppose that the number of individuals and the number of genetic iterations are close to n, then the total computational time necessary to determine the chromosomes' feasibility is $O(n^4)$ where the hidden integer constant may assume a very large value (depending upon the number of genetic evolutions).

Even though this time is polynomial, it is too high to build a canonical genetic algorithm that is sequential in nature. The high computational complexity thus serves as further motivation on the use of a distributed/parallel evolutionary approach.

Once the chromosome template has been defined, the crossover and mutation operators, exploited by the islands to evolve their subpopulation portion, are introduced. Proposed genetic algorithm uses a typical one-point crossover applied with probability $P_{\text{crossover}} = 1/\text{Population Size}$ and, at same time, the algorithm uses a classical mutation operator applied with probability $P_{\text{mutation}} = 1/n$. Our proposal exploits an iterative approach to chose the crossover and mutation point in order to minimize the probability to generate unfeasible chromosomes. In particular, the algorithm generates the crossover point i only if $c[i] \neq c[i + 1]$; in the same way the algorithm generates the mutation point i only if $c[i - 1] \neq c[i]$ *or* $c[i] \neq c[i + 1]$.

The proposed parallel algorithm enables the computation of a feasible solution in a rapid way. Moreover, the solution quality is improved with respect the sequential evolutionary approaches because our proposed approach has improved the speciation level of the population using a hierarchical distribution. However, in order to further refine the quality, the supervisor can improve the parallel genetic solution by means of an appropriate local search process. In particular, each genetic host computes a sub-optimal population through the Gaussian function presented in the previous section and, successively, the supervisor merges the distributed sub-populations and, iteratively, applies a local refinement on chromosomes composing the whole population.

The first step is to define the so-called neighborhood, i.e., a set of feasible solutions that are close to the solution computed by the parallel genetic approach know as L. In particular, the neighborhood contains solutions obtained by adding, deleting or replacing a concept from L. The local search operator exploits the neighborhood to improve the fitness of the current sub-optimal solution.

Formally, let $L = (l_1, l_2, l_3, \ldots, l_n) \in \{1, \ldots, m\}^n$ be a current feasible solution coming from the genetic islands. Clearly, an $(n + 1)$-tuple $(l_1, l_2, \ldots, l_{i-1}, l_i, l'_i, l_{i+1}, \ldots, l_n) \in \{1, \ldots, m\}^{n+1}$ obtained by adding a concept l'_i to the n-tuple $L \in \{1, \ldots, m\}^n$ is not a feasible solution because it does not satisfies chromosome constraints. In the same way an $(n - 1)$-tuple $(l_1, l_2, \ldots, l_{i-1}, l_{i+1}, \ldots, l_n) \in \{1, \ldots, m\}^{n-1}$ obtained by removing a concept l_i, from the n-tuple $L \in \{1, \ldots, m\}^n$ is not a feasible solution. Vice versa a novel solution $L' = (\ldots, l_{i-1}, l'_i, l_{i+1}, \ldots)$ obtained by replacing a concept, l_i with another concept l'_i may be a feasible solution. Different cases have been considered here:

1) $l'_i \neq l_{i-1}, l'_i \neq l_{i+1}$ and $l_{i-1} \neq l_{i+1}$ (l'_i is not present in the remaining part of the chromosome), with $\text{fitness}(L') < \text{fitness}(L)$
2) $l'_i = l_{i-1}$ and $l'_i \neq l_{i+1}$ with $\text{fitness}(L') < \text{fitness}(L)$
3) $l'_i \neq l_{i-1}$ and $l'_i = l_{i+1}$ with $\text{fitness}(L') < \text{fitness}(L)$
4) $l'_i \neq l_{i-1}, l'_i \neq l_{i+1}$ and $l_{i-1} \neq l_{i+1}$ (l'_i is not present in the remaining part of the chromosome), with $\text{fitness}(L') > \text{fitness}(L)$
5) $l'_i = l_{i-1}$ and $l'_i \neq l_{i+1}$ with $\text{fitness}(L') > \text{fitness}(L)$
6) $l'_i \neq l_{i-1}$ and $l'_i = l_{i+1}$ with $\text{fitness}(L') > \text{fitness}(L)$
7) $l'_i \neq l_{i-1}, l'_i \neq l_{i+1}$ and $l_{i-1} = l_{i+1}$ with $\text{fitness}(L') > \text{fitness}(L)$
8) $l'_i \neq l_{i-1}, l'_i \neq l_{i+1}$ and $l_{i-1} = l_{i+1}$ with $\text{fitness}(L') > \text{fitness}(L)$

It is clear that solutions (1), (2), (3) are feasible and optimal solutions, while solutions in cases (4), (5), (6) although are feasible, they are non optimal solutions. Further, the solutions in case (7) is an optimal but unfeasible solution while solutions in case (8) is neither feasible nor optimal. In short, starting from solution L and by applying the replacement of activity l_i with l'_i, we obtain better solution with probability P = number of feasible and optimal solutions/number of solutions = 3/8, i.e., after three refinement steps, the chromosome improves its fitness with high probability.

V. Exploring e-Learning Knowledge: Experimental Results

This paper has proposed an innovative methodology to find an optimal personalized learning activity through the memetic exploration of ontologically defined learning paths. This section is devoted to compare the performance from two different perspectives: a *computational view* where the focus is on the computational effort and the solution quality computed by our

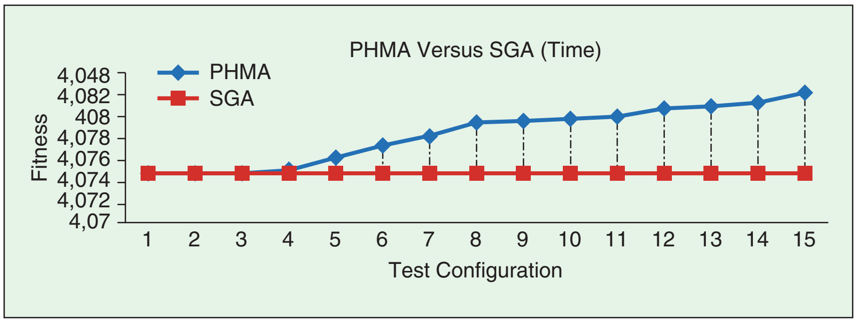


FIGURE 8 PHMA versus SGA–solution quality comparison.

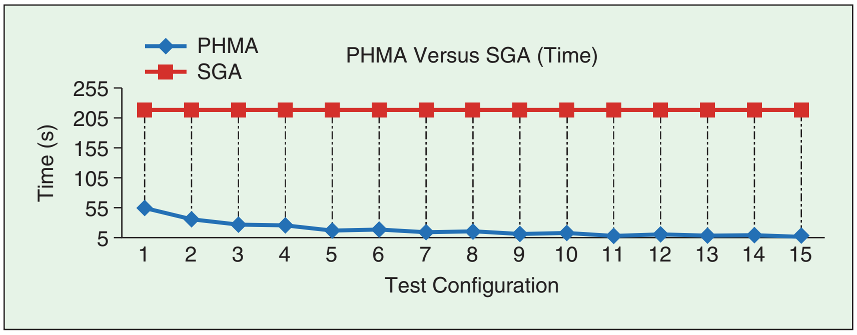


FIGURE 9 PHMA versus SGA–CPU time comparison.

hierarchical memetic algorithm; a *learning view* highlighting the benefits of proposed framework as an add-on to on-line learning. From a computational point of view, the performances of a typical sequential genetic algorithm are compared with our proposal of multi-island memetic approach both in terms of computation time and solution fitness. The parallel simulation were carried out on a cluster of eight AMD Dual Opteron(tm) 250 2.40 GHz hosts, each one equipped with 2 Gb of RAM and running the Microsoft Windows XP x64 operating systems. The simulation of sequential genetic algorithm were carried out on a AMD Opteron 252 2.61GHz with 8 Gb of RAM memory. Our benchmark uses 1) a matrix composed by 40 rows and 1000 columns representing the distances between 40 concepts, composing the personalized learning path, and 1000 learning objects in the distributed repository, and, 2) a vector representing the costs associated with the objects. For sequential genetic algorithm (SGA), the parameters are: Population Size = 2000; Maximum Number of Evolutions = 1000; Crossover Probability = 0.0005; Mutation Probability = 0.025. For the parallel hierarchical memetic algorithm (PHMA), different configurations have been tested for different number of hosts and different number of islands. In detail, let #hosts (with #hosts = 1, . . . , 8) be the number of computer hosting the genetic islands and let #islands (with islands ={2, 3, 4, 5, 6, 7, 8, 10, 12, 14, 16}) be the number of genetic islands used in the systems, then:

$$\begin{aligned}\text{Population Size} &= 2{,}000\\ \text{Island Population Size} &= \frac{2{,}000}{\#\text{islands}}\\ \text{Maximum Number of Evolutions} &= 1{,}000.\end{aligned}$$

Moreover, the Gaussian function exploits the following parameters to select the best chromosomes from island subpopulations: $\mu = 40.783$, $\sigma = 0.01$, $\eta = 0.2$; these values have been computed experimentally by solving the proposed problem through a simple sequential genetic algorithm using a low number of initial chromosomes and search generations. In order to evaluate the algorithms, several criteria measuring the solution quality and computation time have been adopted. *CPU time* represents the average of computation time expressed in milliseconds upon the algorithm termination. *Average fitness* is the average fitness value of the solutions obtained for all the genetic algorithms (sequential and parallel) during a given test configuration. *Best fitness* is the fitness of the best solution obtained among all simulation runs and *gap* is the difference between the fitness the best found solution and the fitness of best known solution (in our case 40.7464) of the benchmark problem. To compare the PHMA and SGA algorithms considering both solution quality and computational time, two additional parameters have been considered: PHMA vs SGA (time) and PHMA vs SGA (Fitness). The former measures the percentage improvement in CPU time; the latter indicates the percentage improvement in solution quality. Our results show that the PHMA algorithms are capable of computing a fitness value comparable with the fitness value computed by a simple SGA but remarkably reducing the computational time. Figs. 8 and 9 show the performance comparisons between PHMA and SGA. Though they are not validated through a statistical method, experimental results have been computed several times in order to obtain an accurate estimation of computed fitness values.

The algorithms were implemented using Java programming language. JGAP (http://jgap.sourceforge.net/) library has been exploited to used to provide the genetic operators features, whereas, the Fracture (http://kccoder.com/fracture/) library has been used distributing the multi-island memetic algorithm on the different processor cores. All communications between supervisor and genetic agents have been implemented through the Java Remote Method Invocation (RMI) platform.

An on-field experimentation was performed to demonstrate the benefits of our optimization system by implementing a plug-in for a commercial Learning Management System named IWT. IWT has been employed in many contexts as enterprise, universities and schools. IWT supports Web 2.0 by integrating resources, tools and services. Indeed, IWT incorporates a wide set of Web 2.0 tools as e-Portfolio, blogs, podcasts, wikis, social networking, shared areas, RSS feeds, etc.. This characteristic

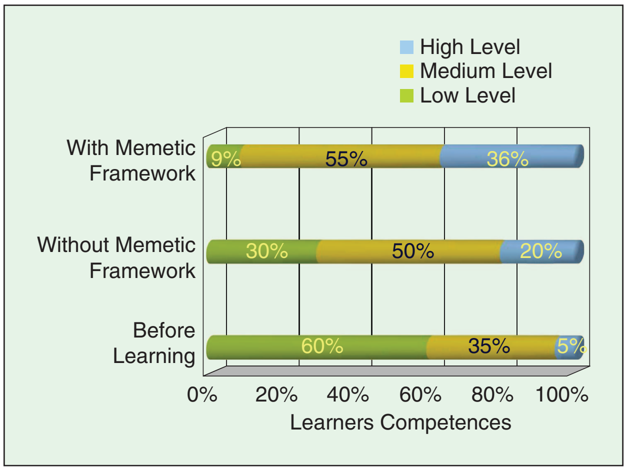


FIGURE 10 Experimentation results on a group of 28 learners with and without the proposed memetic framework.

enables IWT to be a suitable Learning Management System to test our optimization approach.

Our experimentation involved a group of 38 voluntary learners belonging to 7 small and medium enterprises dealing with vocational training on enterprise management. The group of learners was split in two separate sub-groups: a first sub-group composed of 20 learners was enabled to use all e-learning facilities except memetic optimization while a second sub-group composed of 18 learners was enabled to access the whole systems (including hierarchical memetic algorithm). 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). Fig. 10 shows the performances of the two sub-groups; as it can be seen, the progress made by the second group of students is much sharper with respect to the first group [29].

As shown, the proposed ontological/memetic agent-based platform provides an integrated approach towards achieving personalization in e-Learning 2.0 environments by exploiting an advanced knowledge exploitation and exploration based on an innovative computational intelligence area such as the memetic computing.

VI. Conclusions

The proposed ontological/memetic distributed platform provides an integrated approach towards achieving personalization in e-learning environments. Our proposal enhances significantly the overall system in terms of flexibility and efficiency while it introduces a high degree of agents and e-learning platforms interoperability utilizing ontology-based representation. The presented agent platform can support various personalization levels (intended as most fitting sequences of learning activities able to maximize the understanding level of learners with respect to specific learning objectives) exploiting machine-understandable representations of educational domains and learners' characteristics. In particular, aforesaid aims have been achieved through the joint use of different methodologies and techniques: ontological representation, user modeling techniques, graph algorithms. Moreover, we have shown how the ontological representation of e-learning environments allows for modeling a PLP whose solution defines an optimal learning experience. The computational intelligence methodologies and, in particular, the memetic computing methodology has been exploited to solve the e-Learning PLP an efficient manner that allows system designers to realize an efficient in-time learning environment.

References

[1] T. O'Reilly. Design patterns and business models for the next generation of software [Online]. Available: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
[2] M. Mebner, "E-Learning 2.0 = e-Learning 1.0 + Web 2.0?," in *Proc. 2nd Int. Conf. Availability, Reliability and Security 2007, ARES 2007*, pp. 1235–1239.
[3] D. Dagger, A. O'Connor, S. Lawless, E. Walsh, and V. P. Wade, "Service-oriented e-learning platforms: From monolithic systems to flexible services," *IEEE Internet Comput.*, vol. 11, pp. 28–35, 2007.
[4] D. Dicheva and C. Dichev, "TM4L: Creating and browsing educational topic maps," *Br. J. Educ. Technol.*, vol. 37, pp. 391–404, 2006.
[5] C. S. Lee, Z. W. Jian, and L. K. Huang, "A fuzzy ontology and its application to news summarization," *IEEE Trans. Syst., Man, Cybern. B*, vol. 35, pp. 859–880, 2005.
[6] C. S. Lee, M. H. Wang, and J. J. Chen, "Ontology-based intelligent decision support agent for CMMI project monitoring and control," *Int. J. Approx. Reason.*, vol. 48, pp. 62–76. 2008.
[7] S. Bechhofer, Y. Yesilada, R. Stevens, S. Jupp, and B. Horan, "Using ontologies and vocabularies for dynamic linking," *IEEE Internet Comput.*, vol. 12, pp. 32–39, 2008.
[8] J. Krarup and P. M. Pruzan, "The simple plant location problem: Survey and synthesis," *Eur. J. Oper. Res.*, vol. 12, pp. 36–57, 1983.
[9] G. G. Yen, "Learning and intelligence," *IEEE Comput. Intell. Mag.*, vol. 4, no. 2, p. 2, 2009.
[10] P. Stone, *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer* (Intelligent Robotics and Autonomous Agents). Cambridge, MA: MIT Press, 2000.
[11] M. Wooldridge, *An Introduction to MultiAgent Systems*. New York: Wiley, 2002.
[12] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agent Multi-Agent Syst.*, vol. 11, no. 3, 2005.
[13] K. A. De Jong, "Evolving intelligent agents: A 50 year quest," *IEEE Comput. Intell. Mag.*, vol. 3, no. 1, pp. 12–17, 2008.
[14] A. Somani and L. Tesfatsion, "An agent-based test bed study of wholesale power market performance measures," *IEEE Comput. Intell. Mag.*, vol. 3, no. 4, pp. 56–72, 2008.
[15] K. Asanovic, R. Bodik, B. C. Catanzaro, J. J. Gebis, P. Husbands, K. Keutzer, D. A. Patterson, W. L. Plishker, J. Shalf, S. W. Williams, and K. A. Yelick. (2006, Dec.). The landscape of parallel computing research: A view from Berkeley. Elect. Eng. and Comput. Sci., Univ. California, Berkeley, Tech. Rep. UCB/EECS-2006-183 [Online]. Available: http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf
[16] R. Dawkins, *The Selfish Gene*. New York: Oxford Univ. Press, 1976.
[17] Y. S. Ong and A. J. Keane, "Meta-Lamarckian learning in memetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 8, no. 2, pp. 99–110, 2004.
[18] J. Tang, M. H. Lim, and Y. S. Ong, "Parallel memetic algorithm with selective local search for large scale quadratic assignment problems," *Int. J. Innovative Comput., Inform. Control*, vol. 2, no. 6, pp. 1399–1416, 2006.
[19] D. Lim, Y. S. Ong, Y. Jin, B. Sendhoff, and B. S. Lee, "Efficient hierarchical parallel genetic algorithms using grid computing," *Future Gener. Comput. Syst.*, vol. 23, no. 4, pp. 658–670, 2007.
[20] G. Albano, M. Gaeta, and S. Salerno, "E-learning: A model and process proposal," *Int. J. Knowl. Learn.*, vol. 2, pp. 73–88, 2006.
[21] R. Koper, "Current research in learning design," *Educ. Technol. Soc.*, vol. 9, pp. 13–22, 2006.
[22] R. Knappe, H. Bulskov, and T. Andreasen, "Perspectives on ontology-based querying," *Int. J. Intell. Syst.*, vol. 23, pp. 739–776, 1998.
[23] N. Guarino, *Formal Ontology in Information Systems*. IOS Press.
[24] T. J. Kopcha and H. Sullivan, "Learner preferences and prior knowledge in learner-controlled computer-based instruction," *Educ. Technol. Res. Develop.*, vol. 3, pp. 265–286, 2008.
[25] G. Albano, M. Gaeta, and P. Ritrovato, "Testing hypotheses in the functional linear model," *Scand. J. Statist.*, vol. 30, pp. 241–251, 2007.
[26] G. Acampora, M. Gaeta, V. Loia, P. Ritrovato, and S. Salerno, "Optimizing learning path selection through memetic algorithms," in *Proc. IEEE Int. Joint Conf. Neural Networks 2008, IJCNN 2008, (IEEE World Congr. Computational Intelligence)*, pp. 3869–3875.
[27] T. Jing, M. H. Lim, and Y. S. Ong, "A parallel hybrid GA for combinatorial optimization using grid technology," in *Proc. IEEE Congr. Evolutionary Computation*, 2003, pp. 1895–1902.
[28] M. H. Lim, T. Jing, and Y. S. Ong, "Diversity-adaptive parallel memetic algorithm for solving large scale combinatorial optimization problems," *Soft Comput.*, vol. 11, pp. 873–888, 2007.
[29] G. Acampora, M. Gaeta, and V. Loia, "Hierarchical optimization of personalized experiences for e-learning systems through evolutionary models," *Neural Comput. Applicat.*, 2009.