Int. J. Engng Ed. Vol. 25, No. 4, pp. 632–645, 2009
Printed in Great Britain.

0949-149X/91 $3.00+0.00

A New Method for Domain Independent Curriculum Sequencing: A Case Study in a Web Engineering Master Program*

LUIS DE MARCOS, ROBERTO BARCHINO, JOSÉ JAVIER MARTÍNEZ and JOSÉ ANTONIO GUTIÉRREZ
Dpto. Ciencias Computación. Universidad de Alcalá, Spain. E-mail: luis.demarcos@uah.es

In e-learning initiatives, sequencing problem concern the arranging of a particular learning unit's set in a suitable order for a particular learner. Sequencing is usually performed by instructors who create a general-public ordered series rather than learner personalized sequences. This paper proposes an innovative intelligent technique for learning object automated sequencing using particle swarms. E-Learning standards are upheld in order to ensure interoperability. Competencies are used to define relations between learning objects within a sequence, so that the sequencing problem turns into a permutation problem and artificial intelligent techniques can be used to solve it. Particle Swarm Optimization (PSO) is one such technique and it has proved to perform well for solving a wide variety of problems. An implementation of PSO for the learning object sequencing problem is presented and its performance in a real scenario is discussed.

Keywords: e-learning; learning object; sequencing; swarm intelligence; Particle Swarm Optimization (PSO)

INTRODUCTION

BRUSILOVSKY envisaged Web-based adaptive courses and systems as being able to achieve some important features including the ability to act as a substitute for teachers' and other students' support, and the ability to adapt to (and so be used in) different environments by different users (learners). These systems may use a wide variety of techniques and methods. Among them, curriculum sequencing technology serves 'to provide the student with the most suitable individually planned sequence of knowledge units to learn and sequence of learning tasks [. . .] to work with' [1]. Curriculum sequencing has its origins in the field of intelligent tutoring systems where it has been extensively studied. Its techniques have later been adopted by the adaptive hypermedia researchers in their broader approach to building personalized (e-learning) systems. So current methods are derived from the adaptive hypermedia field [2] and they rely on complex conceptual models, usually driven by sequencing rules [3, 4]. E-learning traditional approaches and paradigms, which promote reusability and interoperability, are generally ignored, thus resulting in (adaptive) proprietary systems (such as AHA! [5]) and non-portable courseware.

On the other hand, traditional approaches promote standards' usage to ensure interoperability but they lack flexibility, which is in increasing demand. 'In offering flexible [e-learning] programmes, providers essentially rule out the possibility of having instructional designers set fixed paths through the curriculum' [6]. However, offering personalized paths to each learner will impose prohibitive costs to these providers, because the sequencing process is usually performed by instructors. So, 'it is critical to automate the instructor's role in online training, in order to reduce the cost of high quality learning' [7] and, among these roles, sequencing seems to be a priority.

In this paper, an innovative sequencing technique that automates the teacher's role is proposed. E-Learning standards and the learning object paradigm are used in order to promote and ensure interoperability. The learning units' sequences are defined in terms of competencies in such a way that the sequencing problem can be modelled like a classical Constraint Satisfaction Problem (CSP) and Artificial Intelligence (AI) approaches could be used to solve it. Particle Swarm Optimization (PSO) is an AI technique and it has proved to perform well for solving a wide variety of problems. So, PSO is used to find a suitable sequence within the solution space respecting the constraints. In the next, the conceptual model for competency-based learning object sequencing is presented. The PSO approach for solving the problem is then described. The next section presents the results obtained from intelligent algorithm implementation and testing in a real world situation (course sequencing in an online Master in Engineering program), a comparison with other techniques and a discussion

* Accepted 24 February 2009.

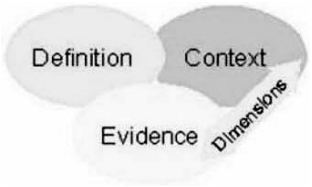

Fig 1. RDCEO competency conceptual model (from [12]).

concerning scalability issues. And, finally, the conclusions are summarized.

COMPETENCY-BASED SEQUENCING

Within e-learning, the learning object paradigm drives almost all initiatives. This paradigm encourages the creation of small reusable learning units called Learning Objects (LOs). These LOs are then assembled and/or aggregated in order to create larger units of instruction (lessons, courses, etc) [8].

LOs must be arranged in a suitable sequence before being delivered to learners. Currently, sequencing is performed by instructors who do not create a personalized sequence for each learner, but instead they create generic courses, which are targeted to generic learner profiles. Then, these sequences are coded using a standard specification to ensure interoperability. The most commonly used specification is SCORM [9]. Courseware that conforms to SCORM's Content Aggregation Model [10] is virtually portable between a wide variety of Learning Management Systems (LMSs). However, SCORM usage hinders the automatic LO sequencing due to its system-centred view. Other metadata-driven approaches offer better possibilities, i.e. just LO metadata will enable automatic sequencing process to be performed, and the appropriate combination of metadata and competencies will allow personalized and automatic content sequencing. This section describes how to overcome these problems by defining a conceptual data model for learning object sequencing through competencies.

Competency definition

As for many other terms, there are a wide variety of definitions that try to catch the essence of the word competency in the e-learning environment. The confusion has even been increased by the work developed, often independently, in the three main fields that are currently primarily concerned with competencies, namely, pedagogy, human resources management and computer science. Anyway, we consider competencies to be 'multidimensional, comprising knowledge, skills and psychological factors that are brought together in complex behavioural responses to environmental cues' [11]. This definition emphasizes that competencies are not only knowledge but a set of factors and that competencies are employed (brought together) in real or simulated contexts (or environments). Conceptual models for competency definitions are also used to consider this multidimensionality. As an example, RDCEO specification [12] describes a competency as a four-dimensional element (Fig. 1).

The competency 'Definition' is the record that contains general information about the competency. Each competency can be exhibited in one or more different 'Contexts'. And a set of factual data must be used to 'Evidence' that an individual has or has not acquired a particular competency. Finally, 'Dimensions' are used to relate each context with its particular evidence and to store relation information such as the proficiency level.

Some e-learning trends (RDCEO have just been mentioned) are trying to formalize competency definitions. It is worth quoting the following specifications:

- IMS 'Reusable Definition of Competency or Educational Objective' (RDCEO) specification [13];
- IEEE Learning Technology Standards Committee (LTSC) 'Standard for Learning Technology—Data Model for Reusable Competency Definitions' specification [14];
- HR-XML Consortium 'Competencies (Measurable Characteristics) Recommendation' [15];
- CEN/ISSS 'A European Model for Learner Competencies' workshop agreement [16].

All these specifications offer their own understanding of what a competency is (i.e. the definition of competency) plus a formal way to define competencies (i.e. competency definitions) so that they can be interchanged and processed by machines. A deeper analysis of these recommendations shows that, although they do not present great differences in their own definitions of competency, great dissimilarities arise when the information that must conform to a competency definition are examined. In this way, it could be said that IMS and IEEE specifications are minimalist recommendations that define a small set of fields that the competency definitions should contain (in fact, only an identifier and a name are required for a conformant record). Deeper definitions of some dimensions that concern competencies (namely evidence and context) are left without specification or are left free to the developers' interpretation. On the other hand, HR-XML specification provides competency users with a huge set of entities, fields and relations that they must fulfil in order to get conformant competency records (although many of them are also optional).

For the purpose of our study we just need a universal way to define, identify and obtain access to competency definitions and that is exactly what the RDCEO specification offers. Moreover, RDCEO is also the oldest specification and is therefore the most used (and the most criticized). These factors lead us to employ RDCEO records for our competency definitions. Code fragment in

```
<?xml version="1.0" encoding="utf-8"?>
<rdceo xsi:schemaLocation="http://www.imsglobal.org/xsd/imsrdceo_rootv1p0
imsrdceo_rootv1p0.xsd  http://www.w3.org/XML/1998/namespace xml.xsd"
xmlns="http://www.imsglobal.org/xsd/imsrdceo_rootv1p0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   <identifier>
     http://www.uah.es/cc/comps/CompsTaxon.xml#1IntroWeb
   </identifier>
   <title>
      <langstring xml:lang="es">
        Introducción a la Web, Internet y a los Sistemas Distribuidos
      </langstring>
      <langstring xml:lang="en">
        Web, Internet and Distributed Systems Introduction
      </langstring>
   </title>
   <description>
      <langstring xml:lang="en">
        A basic understanding of how the Web and the Internet work, and what
are their main protocols, tools and applications. It also comprises a basic
understanding and knowledge of the basic elements that take part in the
communication process: their names, purposes and basic function.
      </langstring>
   </description>
   <definition>
     <model>UAH CC Competency WG</model>
     <statement statementname="Performance">
       <statementtext>
         <langstring xml:lang="en">
           Understands how the Web and Internet works, and what are their main
protocols, tools and applications. Understands and demonstrates its knowledge
about the basic elements that take part in the communication process: their
names, purposes and basic function.
         </langstring>
       </statementtext>
     </statement>
     <statement statementname="Conditions">
       <statementtext>
         <langstring xml:lang="en">
           Has access to the literature and resources about the subject
         </langstring>
       </statementtext>
     </statement>
     <statement statementname="Criteria">
       <statementtext>
         <langstring xml:lang="en">
           Demonstrates and understanding of the concepts by verbally
summarizing them, and by sketching and describing all the elements that take
part in the communication process of a proposed practical situation (as for
example a web page request from a web browser or file request from a FTP
client)
         </langstring>
       </statementtext>
     </statement>
   </definition>
</rdceo>
```

Fig. 2. Code 1. Sample competency record.

Fig. 2 shows a sample RDCEO competency record.

Competencies for Interoperable Learning Object Sequencing

According to RDCEO and IEEE nomenclature, a competency record is called a 'Reusable Competency Definition' (or RCD). RCDs can be attached to LOs in order to define their prerequisites and learning outcomes. We have used this approach to model LO sequences. By defining a competency (or a set of competencies) as an LO outcome, and by identifying the same competency as the prerequisite for another LO (Fig. 3), a constraint between the two LOs is established so that the first LO must precede the second one in a valid sequence.

Meta-Data (MD) definitions are attached to LOs, and, within those definitions, references to competencies (prerequisites and learning outcomes) are included. LOM [17] records have been used for specifying LO Meta-Data. LOM element 9, 'Classification', is used to include competency references as recommended in [12, 18]. So, LOM element 9.1, 'Purpose', is set to 'prerequisite' or 'educational objective', chosen from the permitted vocabulary for this element; and LOM element 9.2 'Taxon Path', including its sub-elements, is used to reference to the competency. Note that more than one Classification element can be included in one single LO in order to specify more than one prerequisite and/or learning outcome. In code fragment 2 (Fig. 4) is shown a sample LO metadata record that holds two competency references: a prerequisite relation and a learning outcome relation. Competency references are showed in bold.

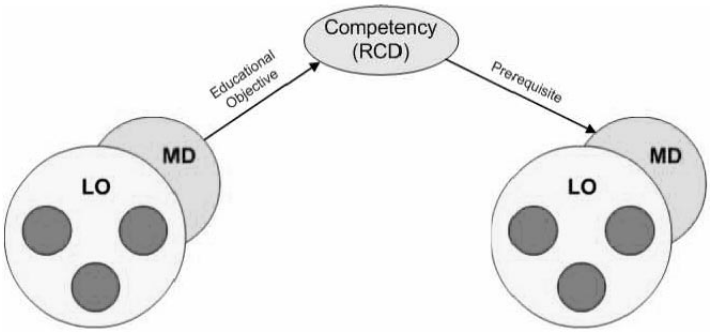

Fig. 3. LO sequencing through competencies.

```
<?xml version="1.0" encoding="iso-8859-1"?>
    <lom:lom xmlns:lom="http://ltsc.ieee.org/xsd/LOM">
      <lom:general>
        <lom:title>
          <lom:string language="es">HTML</lom:string>
        </lom:title>
        <lom:language>es</lom:language>
        <lom:description>
          <lom:string language="es">Visión general sobre el lenguaje de
marcado de páginas Web HTML y sus componentes principales.</lom:string>
        </lom:description>
      </lom:general>
      <lom:lifeCycle>
        <lom:version>
          <lom:string language="es">1.0</lom:string>
        </lom:version>
        <lom:contribute>
          <lom:date>
            <lom:dateTime>2007-01-10</lom:dateTime>
          </lom:date>
        </lom:contribute>
      </lom:lifeCycle>
      <lom:metaMetadata>
        <lom:language>es</lom:language>
        <lom:metadataSchema>LOMv1.0</lom:metadataSchema>
        <lom:metadataSchema>SCORM_CAM_v1.3</lom:metadataSchema>
      </lom:metaMetadata>
      <lom:educational>
        <lom:difficulty>
          <lom:value>easy</lom:value>
        </lom:difficulty>
        <lom:typicalLearningTime>
          <lom:duration>PT50H</lom:duration>
        </lom:typicalLearningTime>
        <lom:language>es</lom:language>
      </lom:educational>
      <lom:classification>
        <lom:purpose>prerequisite</lom:purpose>
        <lom:taxonPath>
          <lom:source>
            <lom:string language="es">
              http://www.uah.es/cc/comps/CompsTaxon/
            </lom:string>
          </lom:source>
          <lom:id>1IntroWeb</lom:id>
        </lom:taxonPath>
      </lom:classification>
      <lom:classification>
        <lom:purpose>educational objective</lom:purpose>
        <lom:taxonPath>
          <lom:source>
            <lom:string language="es">
              http://www.uah.es/cc/comps/CompsTaxon/
            </lom:string>
          </lom:source>
          <lom:id>3HTML</lom:id>
        </lom:taxonPath>
      </lom:classification>
    </lom:lom>
```

Fig. 4. Code 2. Sample LO metadata record containing competency references.

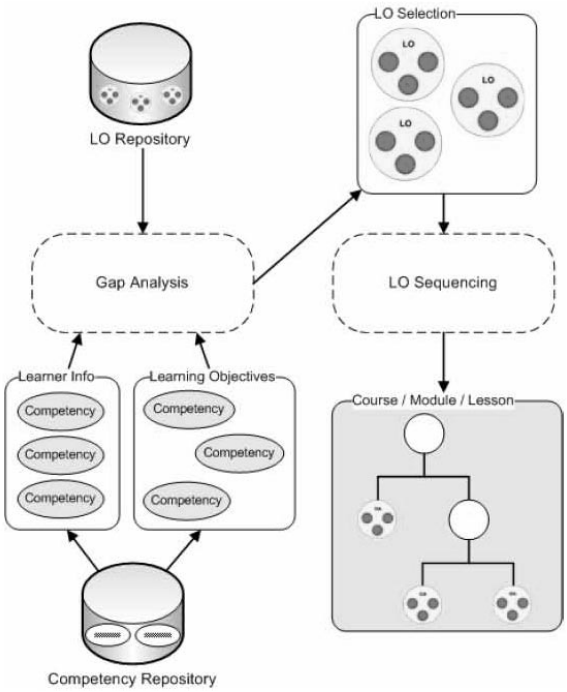

Fig. 5. Competency-driven content generation model.

Simple metadata (i.e. LOM records) are enough to model the LOs' sequences in a similar way. Then, why use competencies? Competency usage is encouraged, in addition to its usefulness for modelling prerequisites and learning outcomes, the competencies are also useful for modelling learners' current knowledge as well as the expected outcomes of the learning initiatives (future learner knowledge).We are proposing a wider framework (Fig. 5) in which learner (user) modelling is done in terms of competencies, which are also used to define the expected learning outcomes from a learning program. Both sets of competencies constitute the input for a gap analysis process. This process performs a search in local and/or distributed remote repositories in order to identify the set of learning objects that fill the gap between the learner's current knowledge and the learning objectives. The gap analysis process returns a set of unordered LOs that must be assembled and structured in a comprehensive way, so that basic units (LOs) are presented to the learner previous to the advanced lessons. These actions will be performed by the LO sequencing process depicted in Fig. 5.

COMPETENCY-BASED INTELLIGENT SEQUENCING

Given a random LOs' sequence modelled as described above (with competencies representing LOs' prerequisites and learning outcomes), the question of finding a correct sequence can be envisaged as a classical Constraint Satisfaction Problem (CSP). In this way, the solution space comprises all possible sequences (*n!* will be its size, total number of states, for *n* LOs), and a (feasible) solution is a sequence that satisfies all established constraints. The LO permutations inside the sequence are the operations that define transitions between states. So we face a permutation problem, which is a special kind of CSP. PSO is a stochastic population-based computing technique that can be used to solve CSP problems (among other kind of problems). This section presents a mathematical characterization of the learning object sequencing problem so that a PSO implementation can be formally specified. This PSO implementation is then presented and some modifications of the original algorithm are proposed.

Mathematical characterization

According to Tsang [19] a CSP is triple (X, D, C) where $X = \{x_0, x_1, \ldots, x_{n-1}\}$ is a finite set of variables, D is a function that maps each variable to its corresponding domain $D(X)$, and $C_{i,j} \subset D_i \times D_j$ is a set of constraints for each pair of values (i, j) with $0 \leq i < j < n$. To solve the CSP is to assign all variables x_i in X a value from its domain D, such that all constraints are satisfied. A constraint is satisfied when $(x_i, x_j) \in C_{i,j}$, and (x_i, x_j) is said to be a valid assignment. If $(x_i, x_j) \notin C_{i,j}$ then the assignment (x_i, x_j) violates the constraint.

If all solutions from a CSP are permutations of a given tuple then it is said that the problem is a permutation CSP or PermutCSP. A PermutCSP is defined by a quadruple (X, D, C, P) where (X, D, C) is a CSP and $P = \langle v_0, v_1, \ldots, v_{n-1} \rangle$ is a tuple of $|X| = n$ values. A solution S of a PermutCSP must be a solution of (X,D,C) and a complete permutation of P.

The learning object sequencing problem could be modelled as a PermutCSP. For example, considering five learning objects titled 1, 2, 3, 4 and 5, the PermutCSP which only solution is the set $S = \{1, 2, 3, 4, 5\}$ (all learning objects must be ordered) can be defined as:

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$
$$D(X_i) = \{1, 2, 3, 4, 5\} \; \forall \; x_i \in X$$
$$C = \{x_{i+1} - x_i > 0 : x_i \in X, I \in \{1, 2, 3, 4\}\}$$
$$P = \langle 1, 2, 3, 4, 5 \rangle$$

As it will be demonstrated later, a good definition of the constraint set C critically affects the solving algorithm performance and even its completeness.

Particle Swarm Optimization

Particle Swarm Optimization (PSO) is an evolutionary computing optimization algorithm. PSO mimics the behaviour of social insects like bees. A randomly initialized particle population (states) flies through the solution space sharing the information they gather. Particles use this information to adjust dynamically their velocity and cooperate towards finding a solution. Best solution found: (1) by a particle is called *pbest*, (2) within a set of neighbour particles is called *nbest*, (3) and within

```
initialize the population
do {
  for each particle X {
    set newfitness = fitnessValue(X)
    if (newfitness > P_pbest)
      set P_pbest = X
  }
  P_nbest = particle with the best fitness value of all the topological
neighbour particles
  for each particle {
    Calculate new velocity as
      V_new=w x V_old + c1 x rand () x (P_pbest - X) + c2 x rand() x (P_nbest- X)
    Update particle position
      X = X + V_new
  }
} until termination criterion is met
```

Fig. 6. Code 3. PSO procedure pseudo-code.

the whole swarm is called *gbest*. *gbest* is used instead of *nbest* when a particle takes the whole population as its topological neighbours. The goodness of each solution is calculated using a function called the fitness function. A basic PSO procedure, adapted from [20], is showed in code fragment 3 (Fig. 6). PSO has been used to solve a wide variety of problems [21].

$\bar{X}$ represents the value of the particle that is currently being computed. $\bar{P}_{\text{pbest}}$ and $\bar{P}_{\text{nbest}}$ are *pbest* and *nbest* values respectively. They are computed for each particle and they are represented as vectors, along with $\bar{X}$, because they usually encode a string of positions (array of values) that represent a multidimensional space. $\bar{V}_{\text{new}}$ is a temporary variable used to compute the new velocity vector. It is an array with the same length of $\bar{X}$ because a velocity value must be computed for each dimension. *w*, *c1* and *c2* are the algorithm parameters, and they are all used to compute the new velocity. *w* is known as the 'inertial weight' and it is a number (from 0.0 to 1.0) that determines to what extent the particle is moved toward the new velocity or remains along its previous course. *c1* and *c2* are known as the 'learning rates' and they determine the relative pull of *pbest* and *nbest* in the velocity computation. Parameter values (*w*, *c1* and *c2*) should be careful selected for each problem. *fitnessValue( )* is a function that returns the fitness function value for one particle. It is called just once per iteration because it usually requires a long computation time. And, finally, *rand( )* is a function that returns a random number between 0 and 1. Each instance of *rand( )* in the algorithm represents a new call to the function, i.e. a new random number is computed and returned.

Original PSO [22, 23] is intended to work on continuous spaces, and velocity is computed for each dimension $x_i \in \bar{x}$. The particles' initial position and initial velocity are randomly assigned when the population (swarm) is initialized. A discrete binary version of the PSO was presented in [24]. This version uses the concept of velocity as a probability of changing a bit state from zero to one or vice versa. A version that deals with permutation problems was introduced in [20]. In this latter version, velocity is computed for each element in the sequence, and this velocity is also used as a probability of changing the element but, in this case, the element is swapped, establishing its value to the value in the same position in *nbest*. The velocity is updated using the same formula for each variable in the permutation set ($x_i \in X$), but it is also normalized to the range 0 to 1 by dividing each x_i by the maximum range of the particle (i.e. maximum value of all $x_i \in X$). The mutation concept is also introduced in this permutation PSO version; after updating each particle's velocity: if the current particle is equal to *nbest* then two randomly selected positions from the particle sequence are swapped. In [20] it is also demonstrated that permutation PSO outperforms genetic algorithms for the N-Queens problem. So we decided to try PSO, before any other technique, for LO sequencing problem.

Each particle shares its information with a, usually fixed, number of neighbour particles to determine the *nbest* value. Determining the number of neighbour particles (the neighbour size) and how the neighbourhood is implemented has been a subject of in-depth research in an area that has been called sociometry. Topologies define structures that determine neighbourhood relations, and several of them (ring, four cluster, pyramid, square and all topologies) have been studied. It has been proved that fully informed approaches outperform all other methods [25]. The fully informed approach prompts using 'all' topology and a neighbourhood size that is equal to the total number of particles in the swarm. This means that every particle is connected to all other particles when *nbest* values are calculated, hence *gbest* is always equal to *nbest*. So, for this case (a fully-informed approach), *nbest* is not longer computed because *gbest* replaces it and only one copy of the *gbest* value is shared by all the particles in the swarm.

```
initialize the population
do {
  for each particle X {
    set newfitness = fitnessValue(X)
    if (newfitness < P_gbest)
      set P_gbest = X
    if (newfitness < P_pbest)
      set P_pbest = X
    Calculate new velocity as
      V_new = w x V_old + c1 x rand() x (P_pbest - X) + c2 x rand() x (P_gbest -
X)
    Normalize Velocity as
      V_norm = V_new / max (V_new)
    Update particle value
      for each v[i] in V_norm {
        if(rand() < v[i])
          swap X[i] for X[indexOf(X, P_gbest[i])]
      }
    Check Mutation
      if (X = P_gbest) swap two random positions from X
  }
} until termination criterion is met
```

Fig. 7. Code 4. PSO procedure for LO sequencing.

PSO for learning object sequencing

A discrete fully-informed version of the PSO was implemented in order to test its performance for solving the LO sequencing problem. Code fragment 4 (Fig. 7) shows the basic procedure for LO sequencing pseudo code. Several other issues concerning design and implementation have to be decided. In the rest of this section each of these issues is discussed and the selection criteria are explained.

Fitness function. It is critical that one chooses a function that accurately represents the goodness of a solution [26]. In PSO, as in other evolutionary technique algorithms and meta-heuristics search procedures, there is usually no objective function to be maximized. A commonly used fitness function when dealing with CSP problems is a standard penalty function [27]:

$$f(X) = \sum_{0 \le i < j < n} V_{i,j}(x_i, x_j), \qquad (1)$$

where $Vi,j : Di \times Dj \{0,1\}$ is the violation function

$$V_{i,j}(\mathrm{x_i}, \mathrm{x_j}) \begin{cases} 0 & \text{if}(\mathrm{x_i}, \mathrm{x_j}) \in C_{i,j} \\ 1 & \text{otherwise} \end{cases}. \qquad (2)$$

The standard penalty function returns the number of constraints violated, so the PSO objective is to minimize that function (sentence if (new fitness > pBest) was changed to if (new fitness < pBest)). When a particle returns a fitness value of 0, a sequence that satisfies all constraints has been found and the algorithm processing is finished.

This fitness function works well if the constraint set C for the PermutCSP has been accurately defined. In the example presented in the section 'Mathematical Characterization', which represents a 5 LO sequence with only one feasible solution, the restriction set was defined as $C = \{x_{i+1} - x_i > 0: x_i \in X, i \in \{1, 2, 3, 4\}\}$. A more accurate definition will be $C = \{x_i - x_j > 0: x_i \in X, x_j \in \{x_1, \ldots, x_i\}\}$. If we consider the sequence {2, 3, 4, 5, 1} the standard penalty function will return 1 if the first definition of C is used, while the returned value will be 4 if the second definition is used. The second definition is more accurate because it returns a better representation of the number of swaps required to turn the permutation into the valid solution. Moreover, the first definition of C has additional disadvantages because some really different sequences (in terms of its distance to the solution) return the same fitness value. For example sequences {2, 3, 4, 5, 1}, {1, 3, 4, 5, 2}, {1, 2, 4, 5, 3} and {1, 2, 3, 5, 4} will return a fitness value of 1. Fortunately, the accurate constraint definition problem could be solved programmatically. A function that recursively processes all restrictions and calculates the most precise set of restrictions violated by a given sequence was developed and called over the input PSO sequence. The user (instructor, content provider, . . .) will usually define the minimum necessary number of constraints and the system will compute 'real' constraints in order to ensure algorithm convergence, so user obligations are lightened simultaneously.

PSO parameters. One important PSO advantage is that it uses a relatively small number of parameters compared with other techniques such as genetic algorithms. However, much has been written in the literature on the subject of the PSO parameter. Hu *et al.* (2003) established the set of parameters in

such a way that PSO works properly for solving permutation problems. So we decided to follow their recommendations, and the parameters were set as follows: Learning rates (*c1*, *c2*) are set to 1.49445 and the inertial weight (w) is computed according to the following equation:

$$w = 0.5 + (rand()/2), \tag{3}$$

where *rand( )* represents a call to a function that returns a random number between 0 and 1. The population size was set to 20 particles. As the fully informed approach was used, it was not necessary to make any consideration concerning the neighbourhood size.

Initialization. The algorithm receives an initial sequence I as an input. This input is used to initialize the first particle. All other particles are initialized randomly by permuting I. The initial velocity for each particle is also randomly initialized as follows: Each $v_i \in V$ is randomly assigned a random value in the range $\{0,|I|\}$, where $|I|$ is the total number of learning objects in the sequence.

Termination criteria. Agent processing stops when a fitness evaluation of a particle returns 0 or when a fixed maximum number of iterations is reached. So the number of iterations was also defined as an input parameter. It was used as a measurement of the number of calls to the fitness function that were allowed to find a solution. It should be noted that some problems may not have a solution, so setting the number of iterations can avoid infinite computing.

Proposed modifications. During the initial agent development we found that in some situations the algorithm got stuck in a local minimum, and it was unable to find a feasible solution. For that reason, two modifications were envisaged in order to try to improve the algorithm performance for LO sequencing. The first change was to decide randomly whether the permutation of a particle's position was performed from *gbest* or from *pbest* ($p = 0.5$). In the original version all permutations were done regarding *gbest*. The second modification consisted of changing *pbest* and *gbest* values when an equal or best fitness value was found by a particle. In other words all the particle's comparisons concerning *pbest* and *gbest* against the actual state were set to less or equal (<=) because the fitness function is to be minimized. The original algorithm determines that *pbest* and *gbest* only change if a better state is found (comparisons strictly <). Code fragment 5 (Fig. 8) presents the final sequencing algorithm pseudo code that includes these modifications. Changes with respect to the basic procedure are showed underscored.

The underlying idea is to increase the particles' mobility and to avoid a quick convergence to local minimums. Three elements are involved in the new velocity computation: the particle's current inertia, *pbest* position and *gbest* position. And the particle's actual movement (which is indeed a permutation) is determined based on the *gbest* current position. If the final permutation is, instead, sometimes performed towards *pbest* we can have particles exploring different regions of the solution space. Otherwise (if all the permutations are

```
initialize the population
do {
  for each particle {
    set newfitness fitnessValue(X)
    if (newfitness <= P_gbest)
      set P_gbest = X
    if (newfitness <= P_pbest)
      set P_pbest = X
    Calculate new velocity as
      V_new = w x V_old + c1 x rand() x (P_pbest - X) + c2 x rand() x (P_gbest -
X)
    Normalize Velocity as
      V_norm = V_new / max(V_new)
    Update particle value
      for each v[i] in V_norm {
        if(rand() < v[i])
          if(rand() < 0.5)
            swap X[i] for X[indexOf(X, P_pbest[i])]
          else
            swap X[i] for X[indexOf(X, P_gbest[i])]
      }
    Check Mutation
      if (X = P_gbest) swap two random positions from X
  }
} until termination criterion is met
```

Fig. 8. Code 5. Modifications to PSO procedure.

performed towards *gbest*), all the particles could end up exploring the same basin, which could finally take the swarm to a local minimum, which is not a feasible solution. This was the purpose of the first modification. As for the second modification, the proposal prompts a change in the values of *pbest* and *gbest* as many times as possible in order to guide the particles towards new directions (these values are used in the new velocity computation). It should also be noted that we are facing a discrete optimization scenario. This means that we can have many states that return the same fitness value. If *pbest* and (especially) *gbest* does not change for a long time the swarm can stagnate and continue exploring one region that does not contain a feasible solution. If *pbest* and *gbest* values are updated every time that an equal or better fitness value is found, we are constantly forcing the particles' movement into new directions and enlarging the search region. All these modifications were tested later in the results phase.

EXPERIMENTAL RESULTS AND DISCUSSION

Problem statement

The PSO algorithm for LOs sequencing described above was designed and implemented using the object-oriented paradigm. We wanted to test its performance in a real scenario, so a problem concerning course sequencing for a Master in Engineering (M.Eng.) programme in our institution, the Computer Science School from the University of Alcalá in Madrid (Spain), was chosen for the test. The Web Engineering M.Eng, program (Fig. 9) comprises 24 courses (subjects) grouped into the following:

- Basic courses (7) that must be taken before any other (kind of course). There may be restrictions between two basic courses, for example the 'HTML' course must precede the Javascript course,
- 'Itinerary' courses (5) that must be taken in a fixed ordered sequence.
- Compulsory courses (5). There may be restrictions between two compulsory courses.
- Elective courses (7). Additional constraints with respect to any other course may be set.

All courses have an expected learning time that ranges from 30 to 50 hours. They are delivered online using a LMS, namely EDVI LMS [28], and every course has its metadata record. Competency records were created to specify the LOs' restrictions, and LOM metadata records were updated to reflect prerequisite and learning outcome competencies as detailed in the *Competency-based Sequencing* section. A feasible sequence must have 23 LOs satisfying all constraints. Please note that the student must choose just one learning path (Java or .NET) and take all the courses on that path. Students must choose six out of the seven elective courses presented in Fig. 9. The five courses of the non-selected path are also eligible as elective subjects. 56 constraints have been defined among different courses. Just

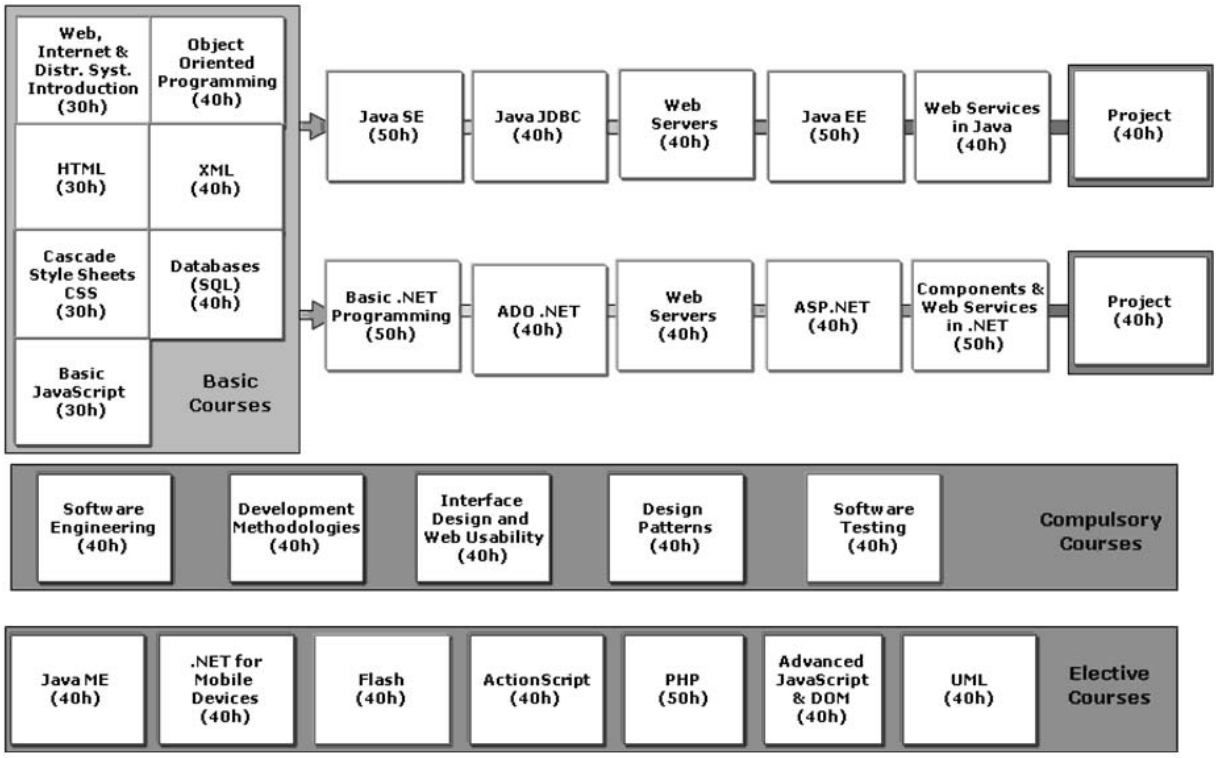

Fig 9. Web Engineering Master Program.

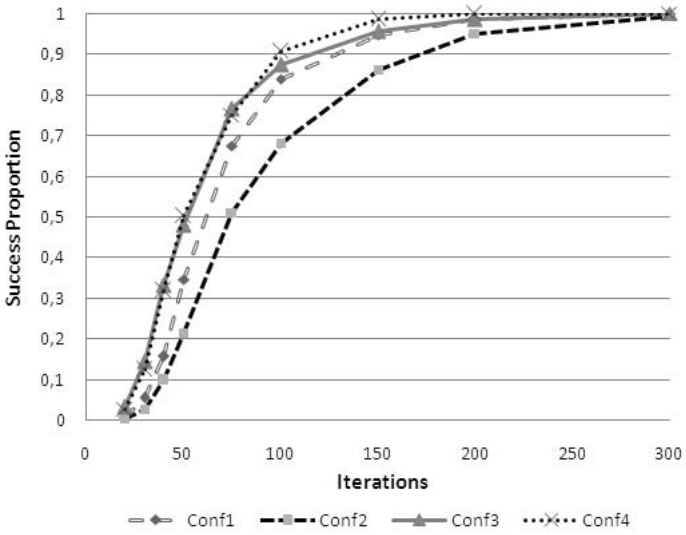

Fig 10. PSO configurations comparison.

to offer an idea, there are four constraints between basic courses, four constraints between compulsory courses, four constraints between itinerary courses and just one constraint in the elective courses. Most of the constraints appear in the relations that take place between courses belonging to different groups: 14 constraints from compulsory courses to basic courses, 7 constraints from itinerary courses to basic courses, 21 constraints from elective courses to basic courses, and one constraint from one elective course to one compulsory course. The graph showing all LOs and constraints is very complex. Figure 9 just represents the constraints between groups of courses. If should be noted that relations between specific courses are not shown in order to avoid confusion in the figure that would make it difficult to read. It is also difficult to calculate the exact number of feasible solutions. Just an estimation has been used; we have estimated that the relation among feasible solutions and total solutions is in the order of 8.9×10^{12}. This number reflects the number of states (non-feasible solutions) for each feasible solution. This means that the chance of a random sequencer selecting a feasible sequence is $1/8.9 \times 10^{12}$, which is quite a reduced probability.

PSO sequencer testing

Once the problem was established, PSO agent parameters were set to test four different configurations that reflect all the possibilities concerning proposed modifications introduced in the *Competency-based Intelligent Sequencing* section. These configurations are:

- Configuration 1. Permutation of the particle position is randomly selected from *gbest* or from *pbest*. Comparison for changing particle *pbest* and *gbest* values is set to less or equal (<=).
- Configuration 2. Permutations from *gbest*/*pbest*. Comparison set to strictly less (<).
- Configuration 3. All permutations are performed from *gbest*. Comparison set to less or equal (<=).
- Configuration 4. Permutations from *gbest.* Comparison set to strictly less (<).

Figure 10 shows the results. Each configuration was run 1000 times allowing 20, 30, 40, 50, 75, 100, 150, 200, 300 and 500 iterations, and the success ratio was observed. From the results, it can be seen that all the configurations converge to a feasible solution, but configuration 4 (original settings) outperform all the others.

Figure 10 also shows that the original settings need less iterations (and so less fitness evaluations) to converge to a feasible solution. This argument is supported by the results in Table 1, which shows the mean number of evaluation function calls required for each configuration to find a solution (1000 runs) if the number of iterations parameter is set to a high enough number (i.e. a number of iterations that ensures a success ratio of 1 for each configuration). Mean times required for each configuration are also presented so that it is possible to get an idea of how long it takes to plan the curriculum for each configuration. We must state that the one objective of our development is to make the curriculum sequencer as scalable as possible (see below) so it is important to consider any difference in performance, even smaller ones, because small differences in performance may turn into large ones if the agent is faced with larger or more difficult scenarios. All the tests were run on a

Table 1. Number of fitness evaluations

	Fitness evaluations	**Time (seconds)**
Configuration 1	1412	14.697
Configuration 2	1817	18.913
Configuration 3	1237	12.875
Configuration 4	1158	12.053

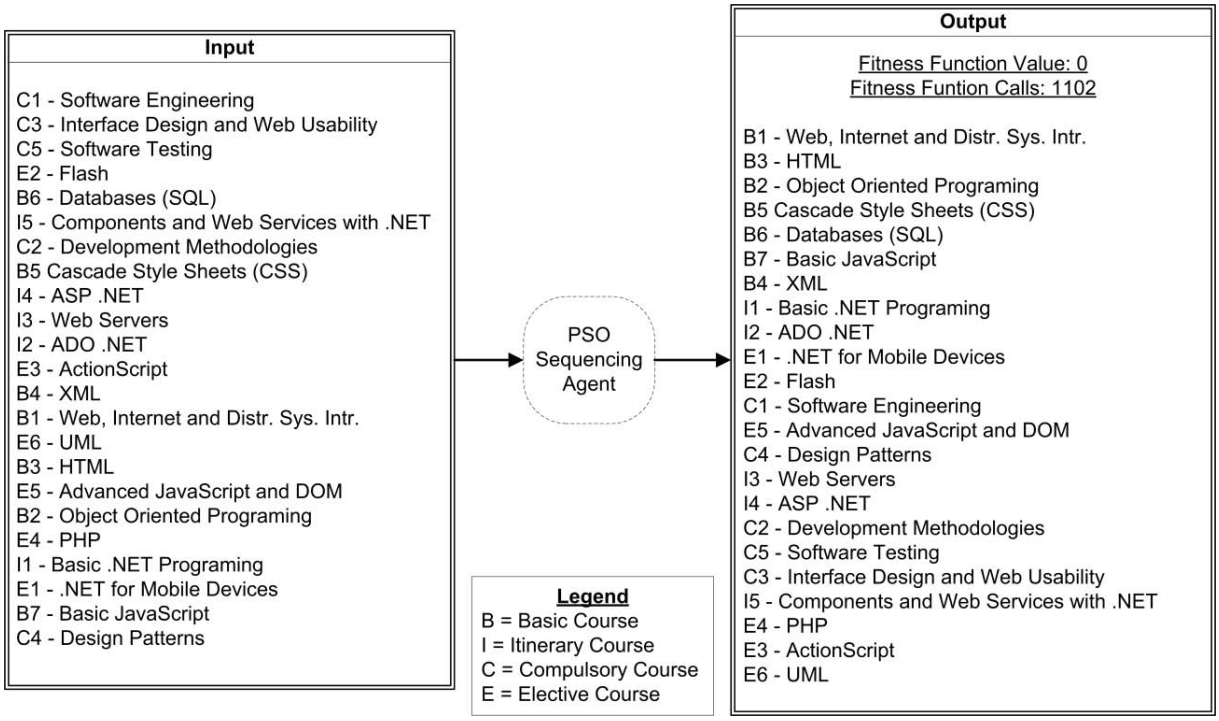

Fig. 11. PSO agent execution example.

computer with a Pentium 4 2.0 GHz processor and 2 GB RAM.

An example of the PSO sequencing agent execution for the test case is shown in Fig. 11. The input is a random sequence of learning objects and the output is a valid sequence (i.e. a sequence that satisfies all restrictions). In the output sequence: (1) all basic courses are placed in the initial positions of the sequence, (2) itinerary courses are properly ordered, and (3) compulsory, itinerary and elective courses are intercalated respecting all constraints. We have supposed that the user takes the .NET path (from Fig. 9) and we have chosen the elective subjects randomly. We plan to control these options later in the user interface. Output is also complemented by the number of fitness function calls required to find the solution.

Comparison with other methods

So far it has just been shown that the algorithm manages to find a solution, however it is questionable whether it performs better or worse than other approaches. A theoretical analysis to compare the PSO sequencer with a random sequencer and with a hill climbing sequencer is presented here to show what the improvement will be over other AI and non-AI techniques. As stated above, the probability of a random solution to be a feasible solution is $1/8.9 \times 10^{12}$. For these kind of approaches, those that employ a heuristic evaluation, the more important consumption of computational resources (time) is related to the fitness evaluation function. We have computed the time that one call to the fitness function consumes and the measure is 10.41 ms (mean time for 65 000 calls was taken). An optimistic evaluation of a random picker will require 4.45×10^{12} evaluations if we estimate that it is necessary to visit half solutions to find one feasible solution ($p = 0.5$). This will represent 4.63×10^{10} seconds, which means that the algorithm will need around 1468 years to find a solution. We are also supposing that no solution is evaluated twice, so a memory to store of at least 4.45×10^{12} solutions will be required and the time required to check that no solution is revaluated must also be computed. This demonstrates that huge solutions spaces cannot be approached in this way if we want to build a computationally feasible solution.

Hill climbing is an optimization technique that belongs to the family of local search techniques. In hill climbing all the neighbours of the current state are evaluated and the best solution is selected as the next value. Then the process is repeated until a feasible solution is found. A random solution is selected to initialize the algorithm. The completeness (ability of the algorithm to find a solution if one exists) of hill climbing is highly dependent on the solution space because the operation stops when the current solution does not find a neighbour with a better fitness value. Solution spaces with ridges (local minima regions) and plateaus (flat parts of the search space in which all the states return the same fitness value) can present a lot of difficulties. Hill climbing works really well when these geographical features are not present in the solution space and it is frequently used in conjunction with other techniques that are specially designed to find the region where a global maximum exist, avoiding local minima. Among its advantages it must be mentioned that it is easy to

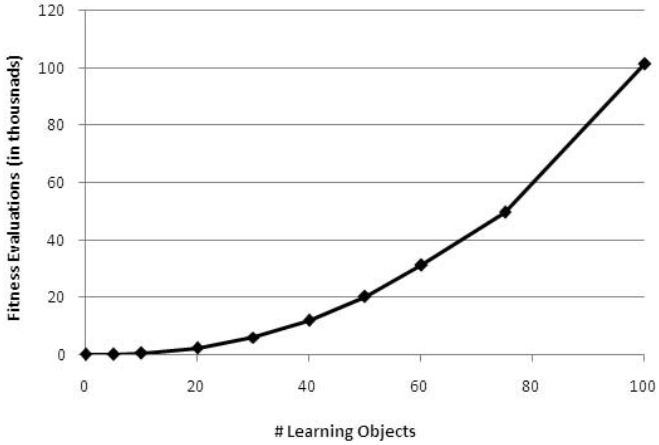

Fig. 12. Number of fitness evaluations required for different number of LOs.

implement, but only if the neighbourhood relation between solutions is clearly stated. Hill climbing is only applicable if every state has a discrete number of neighbours.

Now we need to make a few assumptions to estimate the performance of the hill climber sequencer when planning the curriculum. The first assumption will be that our problem does not contain ridges and plateaus. This is a very important assumption considering that every state will have many potential neighbours. We are also going to suppose that the initial position returns a mean fitness, and that in each iteration of the hill climber is found a new current solution that represents an improvement of two over the fitness value of the previous state. We think that the second assumption is reasonable for this problem because at the beginning of the process the increments will be of more than one unit, but at the final iterations increments of 1 will be more usual. We consider that 2 will be a mean value. For the master program sequencing problem every state has 506 neighbours (23×22) because 23 positions can be chosen for swapping and each position can swap its value with any one of the 22 remaining values. The initial fitness will be 28, 56 are all the constraints and we are considering that one half is the number of them that will be violated. So the hill climber will need $27/2 \times 506 = 6831$ fitness evaluation calls to find a feasible solution. As it has been said that every call to the fitness function requires 10.41 ms it will take approximately 1 minute and 10 seconds for this theoretical hill climber to find a solution. This is 6 time longer than the best PSO sequencer and it will hinder the building of a user interface for the sequencer because response times to the user will be longer.

Scalability issues

The tested scenario may seem to have many feasible solutions so that it is questionable whether PSO would still achieve a good performance in 'challenging' scenarios, so PSO agent was tested in 'more' difficult situations. Test sequences containing 5, 10, 20, 30, 40, 50, 60, 75 and 100 learning objects with only one feasible solution in the solution space were designed. Configuration 4 was used because it showed the best performance for the above test case and unlimited iterations were allowed to find the solution. Fitness evaluation means were observed for 100 runs (Fig. 12) and the mean times required to find the feasible sequence were also calculated (Table 2).

Table 2. Computation times required for different number of LOs

No. of learning objects	Time (mm:ss)
5	0 : 00.707
10	0 : 04.330
20	0 : 23.846
30	1 : 02.859
40	2 : 02.398
50	3 : 29.167
60	5 : 23.780
75	8 : 35.035
100	17 : 33.438

Although fitness evaluations do not increase linearly with the number of learning objects, it should be noted that the learning objects increment entails an exponential explosion of solution space size (remember that solution space size for n learning objects will be $n!$). For example, the solution space with 100 learning objects will be 10^{48} times larger than the solution space with 75 learning objects, but the time required for finding a solution is only about double. In other words, the x-axis could also be interpreted as being the solution space size expressed in a logarithmic scale. Therefore, the intelligent agent also handles reasonably the combinatorial explosion inherent in many AI problems.

CONCLUSIONS

A new method for curriculum sequencing taking standards into account has been presented. The new system is centred on the competency concept.

We note that competencies can be used for modelling user current knowledge as well as learning actions' expected outcomes. These two elements can be the input of a process that automatically creates standard-compliant learning paths that can be used in actual LMSs.

The same (standardized) competency records are used to define constraints between learning objects, so that a sequence of LOs is represented by relations between LOs and competencies. New sequences can then be derived if permutation operations are allowed between LOs in the sequence. Hence the sequencing problem is turned into a permutation problem, and the aim is to find a sequence that satisfies all restrictions expressed in the original model. The PSO for a permutation problem has been extended to an LO sequencing problem. Two envisaged modifications were also tested. Results show that: (1) PSO succeeds as a sequencing method, and (2) the original configuration is the best one. Theoretical comparisons with a random sequencer and with a hill climber approach demonstrate that the PSO sequencer performs better than other AI and non-AI approaches. Finally, some concerns regarding the sequencer scalability have been considered. Different test cases in which the sequencer was faced with the worst possible configurations were run. Results help us to identify what the limit of our approach is and to fine-tune the method to any particular problem.

Further implications arise from the model proposal and from the study conclusions: (1) E-learning standards are promoted. XML records and bindings are used, so elements can be easily interchanged and processed by compliant systems. (2) The Instructor's role is automated, so reducing costs. Sequencing processes work even in complex scenarios where humans face difficulties. Instructors could spend the time saved in performing other activities within the learning process. (3) The model is intended to build domain independent personalized e-learning experiences. We have focused on a Web Engineering master programme but the method will work on any domain if the competency records are defined and the LOs are properly tagged.

Acknowledgments—This research is co-funded by: (1) the University of Alcalá FPI research staff education program, (2) the Spanish Ministry of Industry, Tourism and Commerce PROFIT programme (grants FIT-350200-2007-6, FIT-350101-2007-9, FIT-020100-2008-23, TSI-020302-2008-11) and Plan Avanza programme (grant PAV-070000-2007-103), and (3) Castilla-La Mancha autonomous community under the educational innovation cooperation programme (grant EM2007-004). The authors would also like to acknowledge the support from the TIFyC research group.

REFERENCES

1. P. Brusilovsky, Adaptive and intelligent technologies for web-based education, *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching,* **4**, (1999), pp. 19–25.
2. P. Brusilovsky, Methods and techniques of adaptive hypermedia, *User Modeling and User-Adapted Interaction,* **6**, (1996), pp. 87–129.
3. P. De Bra, G.-J. Houben and H. Wu, AHAM: a Dexter-based reference model for adaptive hypermedia, in *Proceedings of the Tenth ACM Conference on Hypertext and Hypermedia* Darmstadt, Germany, ACM Press, (1999).
4. P. Karampiperis, Automatic learning object selection and sequencing in web-based intelligent learning systems, in *Web-Based Intelligent E-Learning Systems: Technologies and Applications*, M. Zongmin (ed.), Idea Group, London, (2006).
5. P. De Bra, A. Aerts, B. Berden, B. D. Lange, B. Rousseau, T. Santic, D. Smits and N. Stash, AHA! The adaptive hypermedia architecture, in *Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia*, ACM Press, Nottingham, (2003).
6. B. van den Berg, R. van Es, C. Tattersall, J. Janssen, J. Manderveld, F. Brouns, H. Kurvers and R. Koper, Swarm-based sequencing recommendations in e-learning, in *Proceedings 5th International Conference on Intelligent Systems Design and Applications, 2005, ISDA 05.*, Wroclaw, Poland, (2005) pp. 488–493.
7. A. Barr, Revisiting the -ilities: Adjusting the distributed learning marketplace, Again?, *Learning Technology Newsletter,* **8(**January/April), (2006), pp. 3–4.
8. D. A. Wiley, Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy, in *The Instructional Use of Learning Objects*, D. A. Wiley (ed.), (2000).
9. ADL, Shareable Content Object Reference Model (SCORM). The SCORM 2004 Overview, Advanced Distributed Learning (ADL) Initiative, (2004).
10. ADL, Shareable Content Object Reference Model (SCORM). *The SCORM 2004 Content Aggregation Model. Vol. 2005: Advanced Distributed Learning (ADL) Initiative*, (2004).
11. J. Wilkinson, A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM, in *IEEE International Conference on Advanced Learning Technologies, 2001. Proceedings*, (2001) pp. 205–208.
12. IMS, Reusable *Definition of Competency or Educational Objective—Best Practice and Implementation Guide*, IMS Global Learning Consortium, (2002).
13. IMS, Reusable *Definition of Competency or Educational Objective—Information Model*, IMS Global Learning Consortium, (2002).
14. IEEE, Learning Technology Standards Committee (LTSC). *Standard for Learning Technology—Data Model for Reusable Competency Definitions*, IEEE, (2007).
15. HR-XML, *Competencies (Measurable Characteristics) Recommendation*, HR-XML Consortium, (2006).

16. CEN/ISSS, European Model for Learner Competencies, Comité Européen de Normalisation/ Information Society Standardization System (CEN/ISSS), (2006).
17. IEEE, Learning Technology Standards Committee (LTSC). *Learning Object Metadata (LOM)*. 1484.12.1, IEEE, (2002).
18. IEEE, Learning Technology Standards Committee (LTSC). *Standard for Learning Technology—Extensible Markup Language (XML) Schema Definition Language Binding for Learning Object Metadata*. 1484.12.3., IEEE, (2005).
19. E. Tsang, *Foundations of Constraint Satisfaction,* Academic Press, London, (1993).
20. X. Hu, R. C. Eberhart and Y. Shi, Swarm intelligence for permutation optimization: a case study of n-queens problem, in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, Indianapolis, USA, (2003) pp. 243–246.
21. M. G. Hinchey, R. Sterritt and C. Rouff, Swarms and Swarm Intelligence, *Computer,* **40**, (2007), pp. 111–113.
22. R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS 95.*, Nagoya, Japan, (1995), pp. 39–43.
23. J. Kennedy and R. Eberhart, Particle swarm optimization, in *Proceeding IEEE International Conference on Neural Networks*, Perth, WA, Australia, (1995), Vol. 4, pp. 1942–1948.
24. J. Kennedy and R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in *1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation,* (1997), pp. 4104–4108.
25. R. Mendes, J. Kennedy and J. Neves, The fully informed particle swarm: simpler, maybe better, *IEEE Transactions on Evolutionary Computation,* **8**, (2004), pp. 204–210.
26. J. Robinson and Y. Rahmat-Samii, Particle swarm optimization in electromagnetics, *IEEE Transactions on Antennas and Propagation,* **52**, (2004), pp. 397–407.
27. L. Schoofs and B. Naudts, Ant colonies are good at solving constraint satisfaction problems, in *Proceedings of the 2000 Congress on Evolutionary Computation*, La Jolla, CA, (2000), pp. 1190–1195.
28. R. Barchino, J. M. Gutiérrez and S. Otón, An example of learning management system, in *IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005)*, Virtual, (2005) pp. 140–141.

Luis de Marcos has a BSc (2001) and an MSc (2005) in Computer Science from the University of Alcalá, where he is currently a PhD candidate in the Information, Documentation and Knowledge programme. He is member of the IEEE Learning Technology Standard Committee (LTSC). His research interests include competencies and learning objectives definition, adaptable and adaptative systems, learning objects, mobile and ambient learning, and e-learning standarization.

Roberto Barchino has a Computer Sciences Engineering degree from the Polytechnics University of Madrid (2004) and a Ph.D. from the University of Alcalá (2007). Currently, he is Associate Professor in the Computer Science Department of the University of Alcalá and Tutor Professor in the Open University of Spain (UNED). He has been invited to 'Dipartimento di Informatica e Automatica', University of Rome Tre, Rome, Italy, for five months. He is author or co-author of more than 60 scientific works (books, articles, papers and research projects), some of them directly related to Learning Technology. He is also member of the group for the standardization of Learning Technology in the Spanish Official Body for Standardization (AENOR).

José Javier Martínez has a Computer Sciences Engineering degree from the Polytechnics University of Madrid (1993) and a Ph.D. from University of Alcalá (2004). Currently, he is a lecturer in the Computer Science Department of the University of Alcalá. His research interests focus on distributed architectures and tools for e-learning, tutoring systems, learning objects and e-learning standardization.

José Antonio Gutiérrez de Mesa obtained his degree in Computer Science from the Polytechnics University of Madrid (1987), his Documentation degree from the Alcalá University (1996), his Mathematical degree from de University Complutense of Madrid (1997) and his Ph.D. from the Mathematical Department of the University of Alcalá (1999). From September 1992 to April 1994 he worked as a lecturer in the Mathematical Department of Alcalá University, before he joined the Computer Science Department of University of Alcalá as permanent professor. His research interests mainly focus on topics related to software engineering and knowledge representation where he supervises several Ph.D. works in these areas.