5

Competency-based Learning Object Sequencing using Particle Swarms

Luis de Marcos, Carmen Pagés, José Javier Martínez
and José Antonio Gutiérrez
University of Alcalá.
Spain

1. Introduction

Brusilovsky (1999) envisaged Web-based adaptive courses and systems as being able to achieve some important features including the ability to substitute 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 is "to provide the student with the most suitable individually planned sequence of knowledge units to learn and sequence of learning tasks […] to work with". These methods derive from the adaptive hypermedia field (Brusilovsky, 1996) and rely on complex conceptual models, usually driven by sequencing rules (De Bra et al., 1999; Karampiperis, 2006). E-learning traditional approaches and paradigms, that promote reusability and interoperability, are generally ignored, thus resulting in (adaptive) proprietary systems (such as AHA! (De Bra et al., 2003)) and non-portable courseware.
On the other side, traditional approaches promote standards usage to ensure interoperability but they lack of 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" (van den Berg et al., 2005). But offering personalized paths to each learner will impose prohibitive costs to these providers, because 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" (Barr, 2006) and, among these roles, sequencing seems to be a priority.
In this chapter an innovative sequencing technique that automates teacher´s role is proposed. E-Learning standards and the learning object paradigm are encouraged in order to promote and ensure interoperability. Learning units' sequences are defined in terms of competencies in such a way that sequencing problem can be modelled like a classical Constraint Satisfaction Problem (CSP) and Artificial Intelligent (AI) approaches could be used to solve it. Particle Swarm Optimization (PSO) is an AI technique and it has proven with a good performance for solving a wide variety of problems. So, PSO is used to find a suitable sequence within the solution space respecting the constraints. In section 2, the conceptual model for competency-based learning object sequencing is presented. Section 3 describes the PSO approach for solving the problem. Section 4 presents the results obtained

from the intelligent algorithm implementation and testing in a real world situation (course sequencing in an online Master in Engineering program). And finally, in Section 5 conclusions are summarized and future research lines are presented.

2. 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 greater units of instruction (lessons, courses, etc) (Wiley, 2000).
LOs must be arranged in a suitable sequence prior to its delivery 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 (ADL, 2004). Courseware that conforms to SCORM´s Content Aggregation Model is virtually portable among a wide variety of Learning Management Systems (LMSs). Though, SCORM usage hinders the automatic LO sequencing due to its system-centered 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.

2.1 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 nowadays primarily concerned with competencies, namely, pedagogy, human resources management and computer science. Anyway, we consider competencies as "multidimensional, comprised of knowledge, skills and psychological factors that are brought together in complex behavioural responses to environmental cues" (Wilkinson, 2001). This definition emphasizes that competencies are not only knowledge but a set of factors and that competencies are employed (bring together) in real or simulated contexts (or environments). Conceptual models for competency definitions also use to consider this multidimensionality. As an example, RDCEO specification (IMS, 2002a) describes a competency as 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: (1) IMS "Reusable Definition of Competency or Educational Objective" (RDCEO) specification (IMS, 2002b), (2) IEEE Learning Technology Standards Committee (LTSC) "Draft Standard for Learning

Technology - Standard for Reusable Competency Definitions " specification (currently an approved draft) (IEEE, 2008), (3) HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" (HR-XML, 2006) and (4) CEN/ISSS “A European Model for Learner Competencies” workshop agreement (CEN/ISSS, 2006).

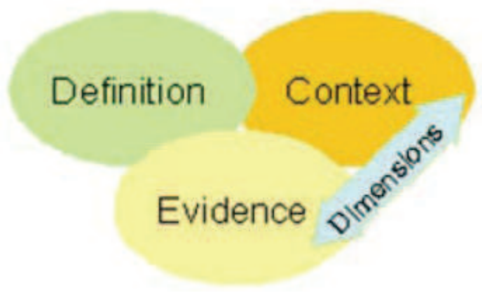

Fig. 1. RDCEO competency conceptual model (from (IMS, 2002a))

Every specification offers its 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 its own definition of competency, great dissimilarities arise when the information that must conform a competency definition are confronted. 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 free to 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 optional too).

For the purpose of our study we just needed a universal way to define, identify and access to competency definitions and that is exactly what RDCEO specification offers. Moreover, RDCEO is also the oldest specification and so the most used (and the most criticized). These factors lead us to employ RDCEO records for our competency definitions. Code fragment 1 shows a sample RDCEO competency record.

```
<?xml version="1.0" encoding="utf-8"?>
<rdceo xsi:schemaLocation="http://www.imsglobal.org/xsd/imsrdceo_rootv1p0"
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="en">
        Web, Internet and Distributed Systems Introduction
      </langstring>
   </title>
</rdceo>
```

Code 1. Sample Competency Record.

2.2 Competencies for interoperable learning object sequencing

According to RDCEO and IEEE nomenclature, a competency record is called 'Reusable Competency Definition' (or RCD). RCDs can be attached to LOs in order to define its prerequisites and its learning outcomes. We have used this approach to model LO sequences. By defining a competency (or a set of competencies) as a LO outcome, and by identifying the same competency as the prerequisite for another LO (fig. 2), 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 (IEEE, 2002) records have been used for specifying LO Meta-Data. LOM element 9, 'Classification', is used to include competency references as recommended in by IMS (2002a). So, LOM element 9.1, 'Purpose', is set to 'prerequisite' or 'educational objective' from among the permitted vocabulary for this element; and LOM element 9.2 'Taxon Path', including its sub-elements, is used to reference 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 it is shown a sample LO metadata record that holds two competency references, a prerequisite relation and a learning outcome relation.

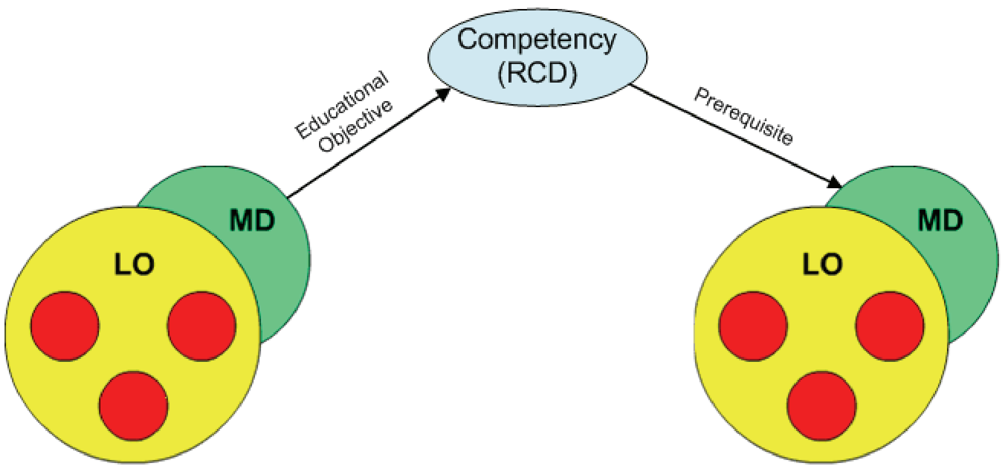

Fig. 2. LO sequencing through competencies

```
<?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="en">HTML</lom:string>
        </lom:title>
        <lom:language>en</lom:language>
        <lom:description>
          <lom:string language="en">HTML Course</lom:string>
        </lom:description>
      </lom:general>
      <lom:lifeCycle>
        <lom:version>
          <lom:string language="en">1.0</lom:string>
        </lom:version>
        <lom:contribute>
          <lom:date>
            <lom:dateTime>2007-01-10</lom:dateTime>
          </lom:date>
        </lom:contribute>
      </lom:lifeCycle>
      <lom:educational>
        <lom:difficulty>
          <lom:value>easy</lom:value>
        </lom:difficulty>
        <lom:typicalLearningTime>
          <lom:duration>PT50H</lom:duration>
        </lom:typicalLearningTime>
        <lom:language>en</lom:language>
      </lom:educational>
      <lom:classification>
        <lom:purpose>prerequisite</lom:purpose>
        <lom:taxonPath>
          <lom:source>
            <lom:string language="en">
              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="en">
              http://www.uah.es/cc/comps/CompsTaxon/
            </lom:string>
          </lom:source>
          <lom:id>3HTML</lom:id>
        </lom:taxonPath>
      </lom:classification>
    </lom:lom>
```

Code 2. Sample LO metadata record containing competency references

Simple metadata (i.e. LOM records) is enough to model LOs' sequences in a similar way. Then, Why use competencies? Competency usage is encouraged, besides its usefulness for modelling prerequisites and learning outcomes, because competencies are also useful for modelling user current knowledge and learning initiatives' expected outcomes (future learner knowledge).We are proposing a wider framework (fig. 3) 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 learner current knowledge and the learning objectives. 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 previously to advanced lessons. These actions will be performed by the LO sequencing process depicted in figure 3.

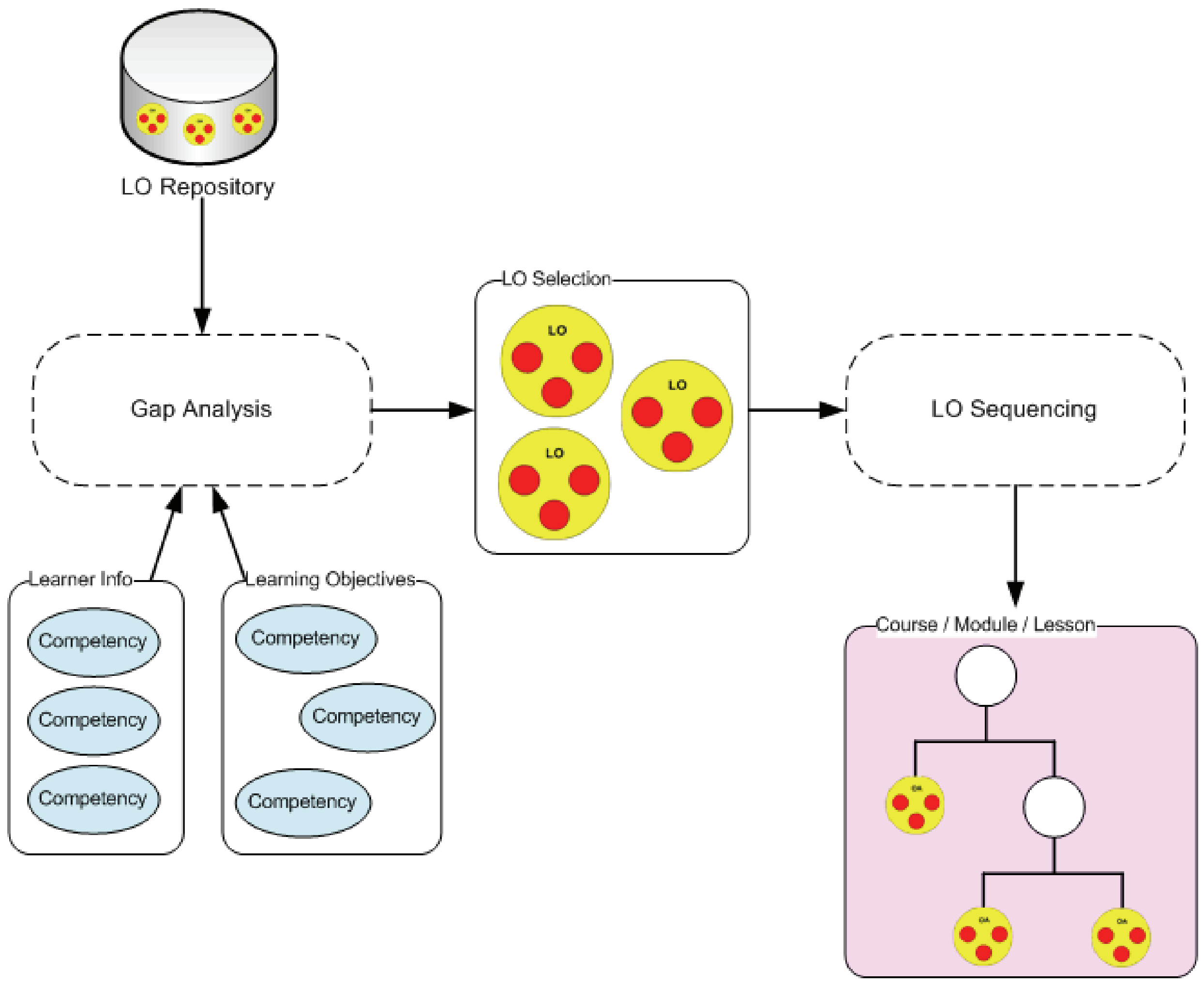

Fig. 3. Competency-driven content generation model

3. 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 artificial intelligent 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. LO permutations inside the sequence are the operations that define transitions among states. So we face a permutation problem, which is a special kind of CSP. PSO is an AI evolutionary 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. Then this PSO implementation is presented and some improvements over the original algorithm are proposed.

3.1 Mathematical characterization

According to (Tsang, 1993) a CSP is a triple *(X,D,C)* where $X = \{x_o, x_1,\ldots,x_{n-1}\}$ is 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*, in such a way that all constraints are satisfied. A constraint is satisfied when $(x_i, x_j) \in C_{i,j}$ and (x_i, x_j) it 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 $P=<v_0, v_1, \ldots, v_{n-1}>$ 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 modeled 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:

$$X = \{x_1, x_2, x_3, x_4, x_5\}$$

$$D\ (X_i) = \{1,2,3,4,5\}\ \forall\ x_i \in X$$

$$C = \{x_{i+1} - x_i > 0 : x_i \in X\ , i \in \{1,2,3,4\}\}$$

$$P = <1,2,3,4,5>$$

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

3.2 Particle swarm optimization

Particle Swarm Optimization (PSO) is an evolutionary computing optimization algorithm. PSO mimics the behaviour of social insects like bees. A random initialized particles' population (states) flies through the solution space sharing the information they gather. Particles use this information to dynamically adjust its 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 the whole swarm is called *gbest*. Goodness of each solution is calculated using a function called fitness function. A basic PSO procedure, adapted from (Hu et al., 2003), is showed in code fragment 3. PSOs have been used to solve a wide variety of problems (Hinchey et al., 2007).
The original PSO (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995) is intended to work on continuous spaces, and velocity is computed for each dimension $x_i \in \bar{x}$. Particles' initial position and initial velocity are randomly assigned when the population (swarm) is initialized. A discrete binary version of the PSO was presented by Kennedy and Eberhart (1997). 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 by Hu et al., (2003). 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*. 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. Hu et al., (2003) have also demonstrated that permutation PSO outperforms genetic algorithms for the N-Queens problem. So we decided to try PSO, before any other technique, for the LO sequencing problem.

```
initialize the population
do {
  for each particle {
    calculate fitness value
    if (new fitness > pBest)
      set pbest = current value
  }
  nbest = particle with the best fitness value of all the topological neighbors
  for each particle {
    Calculate new velocity as
      $\overline{V}_{new}$=w x $\overline{V}_{old}$ + c1 x rand () x ($\overline{P}_{best}$ - $\overline{X}$) + c2 x rand() x ($\overline{P}_{nbest}$- $\overline{X}$)
    Update particle position
      $\overline{X}_{new}$= $\overline{X}_{old}$ + $\overline{V}_{new}$
  }
} until termination criterion is met
```

Code 3. PSO Procedure Pseudo-code

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.

Each particle shares its information with a, usually fixed, number of neighbor particles to determine *nbest* value. Determining the number of neighbor particles (the neighbor size) and how neighborhood is implemented has been a subject of deep research in an area that has been called sociometry. Topologies define structures that determine neighborhood 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 (Mendes et al., 2004). The fully informed approach prompts using an 'all' topology and a neighborhood size equal to the total number of particles in the swarm (i.e. every particle is connected with all other particles when *nbest* values are calculated, hence *gbest* is always equal to *nbest*).

3.3 PSO for learning object sequencing

Discrete full-informed version of the PSO was implemented in order to test its performance for solving the LO sequencing problem. Code fragment 4 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 to choose a function that accurately represents the goodness of a solution (Robinson & Rahmat-Samii, 2004). In PSO, like in other evolutionary techniques algorithms and meta-heuristics search procedures, there is usually no objective function to be maximized. A common used fitness function when dealing with CSP problems is a standard penalty function (Schoofs & Naudts, 2000):

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

where Vi,j : Di x Dj →{0,1} is the violation function

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

```
initialize the population
do {
 for each particle {
  calculate fitness value
  if (new fitness < gBest)
   set gbest = currentValue
  if (new fitness < pBest)
   set pbest = currentValue
  Calculate new velocity as
   $\overline{V}_{new} = w \times \overline{V}_{old} + c1 \times rand() \times (\overline{P}_{pbest} - \overline{X}) + c2 \times rand() \times (\overline{P}_{gbest} - \overline{X})$
  Normalize Velocity as
   $\overline{V}_{norm} = \overline{V}_{new} / \max(\overline{V}_{new})$
  Update particle value
   for each v[i] in $\overline{V}_{norm}$ {
    if(rand() < v[i])
     swap currentValue[i] for indexOf(currentValue, gBest[i])
   }
  Check Mutation
   if (currentValue = gBest) swap two random positions from currentValue
 }
} until termination criterion is met
```

Code 4. PSO Procedure for LO Sequencing

The standard penalty function returns the number of constraints violated, so 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 section 3.1 that 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 $C= \{x_i-x_j>0: x_i \in X, x_j \in \{x_1,\dots,x_i\}\}$. 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. This process was called the 'real' constraint calculator. 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 like genetic algorithms. However, much literature on PSO parameter selection has been written. Among it, 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 parameters were set as follows: Learning rates (*c1*, *c2*) are set to 1.49445 and the inertial weight (*w*) is computed according to the equation (3).

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

where *rand()* represents a call to a function that returns a random number between 0 and 1. Population size was set to 20 particles. As the fully informed was used, it was not necessary to make any consideration concerning the neighborhood 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*. Initial velocity for each particle is also randomly initialized as follows: Each $v_i \in V$ is randomly assigned a value from 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 the number of iterations setting can avoid infinite computing.

Proposed improvements. During the initial agent development we found that in some situations the algorithm got stuck in a local minimum, and it was not able to find a feasible solution. For that reason, two enhancements were envisaged in order to improve algorithm performance for LO sequencing. First improvement 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 improvement was consisted in changing *pbest* and *gbest* values when an equal or best fitness value was found by a particle. In other words all 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 presents the final sequencing algorithm pseudo code that includes these improvements. Changes respecting the basic procedure are showed underscored.

These changes resemble to be quite logical ways for increasing particles' mobility and for avoiding quick convergence to local minimums. And they were tested later in the results phase.

```
initialize the population
do {
 for each particle {
  calculate fitness value
  if (new fitness <= gBest)
   set gbest = currentValue
  if (new fitness <= pBest)
   set pbest = currentValue
  Calculate new velocity as
  $\overline{V}_{new}$ = w x $\overline{V}_{old}$ + c1 x rand() x ($\overline{P}_{pbest}$ - $\overline{X}$) + c2 x ran() x ($\overline{P}_{gbest}$ - $\overline{X}$)
  Normalize Velocity as
  $\overline{V}_{norm}$ = $\overline{V}_{new}$ / max ($\overline{V}_{new}$)
  Update particle value
   for each v[i] in $\overline{V}_{norm}$ {
    if(rand() < v[i])
     if(rand() < 0.5)
      swap currentValue[i] for currentValue[indexOf(currentValue, pBest[i]])
     else
      swap currentValue[i] for currentValue[indexOf(currentValue, gBest[i]])
   }
  Check Mutation
   if (currentValue = gBest) swap two random positions from currentValue
 }
} until termination criterion is met
```

Code 5. Improvements on PSO Procedure

4. Experimental results and discussion

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.) program in our institution, the Computer Science School from the University of Alcalá in Madrid (Spain), was chosen for testing. The (web engineering) M.Eng, program comprises 23 courses (subjects) grouped in:

- Basic courses (7) that must be taken before any other (kind of course). There may be restrictions between two basic courses, for example 'HTML' course must precede 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 (6). 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 (Barchino et al., 2005), and every course has its metadata record. Competency records were created to specify LOs' restrictions, and LOM metadata records were updated to reflect prerequisite and learning outcome competencies as detailed in section 2. A feasible sequence must have 23 LOs satisfying all constraints. The graph showing all LOs and constraints is very complex, and so it is to calculate the exact number of feasible solutions. Some estimations have been used, we have estimated that the relation among feasible solutions and total solutions order is 8,9x10^{12}. This number reflects the number of states (non-feasible solutions) for each feasible solution.

Once the problem was established, PSO agent parameters were set to test four different configurations that reflect all possibilities concerning proposed improvements introduced in Section 3. 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 4 shows the results. Each configuration was run 1000 times allowing 20, 30, 40, 50, 75, 100, 150, 200, 300 and 500 iterations, and the succeed ratio was observed. From the results, it can be seen that all configurations converge to a feasible solution, but configuration 4 (original settings) outperform all others. Figure 4 also shows that original settings need less fitness evaluations. This argument is supported by table 1 results, where it is showed 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 number high enough (i.e. a number of iterations that ensures a success ratio of 1 for each configuration).

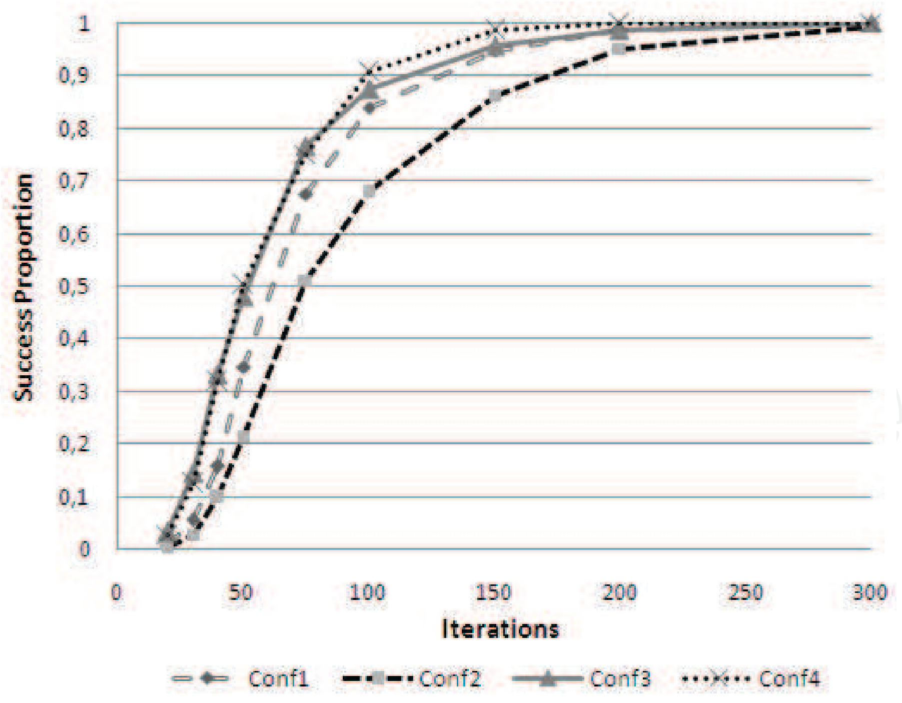

Fig. 4. PSO Configurations Comparison

An example of the PSO sequencing agent execution for the test case is shown in figure 5. 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. Output is also complemented by the number of fitness function calls required to find the solution.

The tested scenario may seem to have many feasible solutions that would make doubtful PSO performance in not-so-kind 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. 6).

Although fitness evaluations does not increase linearly to the number of learning objects, it should be noted that 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 bigger than the solution space with 75 learning objects, but the number of fitness evaluations required for finding a solution is only twice bigger. In other words, X-axis could also be interpreted as the solution space size expressed in a logarithmic scale. Therefore, the intelligent agent also handles reasonably the combinatorial explosion inherent to many AI problems.

	Fitness Evaluations
Configuration 1	1412
Configuration 2	1817
Configuration 3	1237
Configuration 4	1158

Table 1. Number of Fitness Evaluations

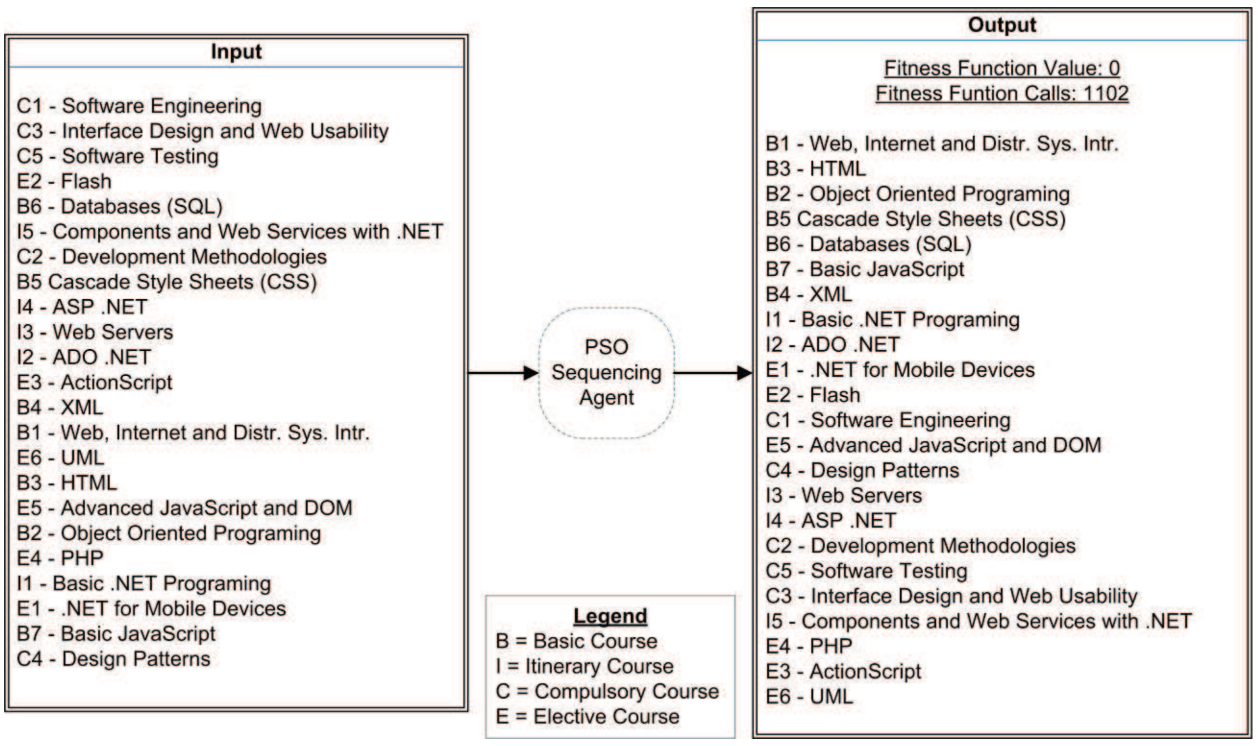
Input
C1 - Software Engineering
C3 - Interface Design and Web Usability
C5 - Software Testing
E2 - Flash
B6 - Databases (SQL)
I5 - Components and Web Services with .NET
C2 - Development Methodologies
B5 Cascade Style Sheets (CSS)
I4 - ASP .NET
I3 - Web Servers
I2 - ADO .NET
E3 - ActionScript
B4 - XML
B1 - Web, Internet and Distr. Sys. Intr.
E6 - UML
B3 - HTML
E5 - Advanced JavaScript and DOM
B2 - Object Oriented Programing
E4 - PHP
I1 - Basic .NET Programing
E1 - .NET for Mobile Devices
B7 - Basic JavaScript
C4 - Design Patterns
PSO Sequencing Agent
Legend
B = Basic Course
I = Itinerary Course
C = Compulsory Course
E = Elective Course
Output
Fitness Function Value: 0
Fitness Funtion Calls: 1102
B1 - Web, Internet and Distr. Sys. Intr.
B3 - HTML
B2 - Object Oriented Programing
B5 Cascade Style Sheets (CSS)
B6 - Databases (SQL)
B7 - Basic JavaScript
B4 - XML
I1 - Basic .NET Programing
I2 - ADO .NET
E1 - .NET for Mobile Devices
E2 - Flash
C1 - Software Engineering
E5 - Advanced JavaScript and DOM
C4 - Design Patterns
I3 - Web Servers
I4 - ASP .NET
C2 - Development Methodologies
C5 - Software Testing
C3 - Interface Design and Web Usability
I5 - Components and Web Services with .NET
E4 - PHP
E3 - ActionScript
E6 - UML

Fig. 5. PSO Agent Execution Example

5. Conclusions

Automated LO sequencing is a recurring problem in the e-learning field that could be undertaken employing models that ensure interoperability and artificial intelligent techniques. The purpose of the study was to design, develop and test a PSO agent that performs automatic LO sequencing through competencies. A model that employs competencies as a mean for defining constraints between learning object has been presented, so that a sequence of LOs is defined by relations among LOs and competencies. New sequences can be derived if permutation operations are allowed between LOs in the sequence. Hence the sequencing problem is turn into a permutation problem, and the aim is to find a sequence that satisfies all restrictions expressed in the original model. The PSO for permutation problem has been extended to LO sequencing problem. Testing two envisaged improvements was also performed. Results show that: (1) PSO succeeds in solving the problem, and (2) the original configuration is the best one.

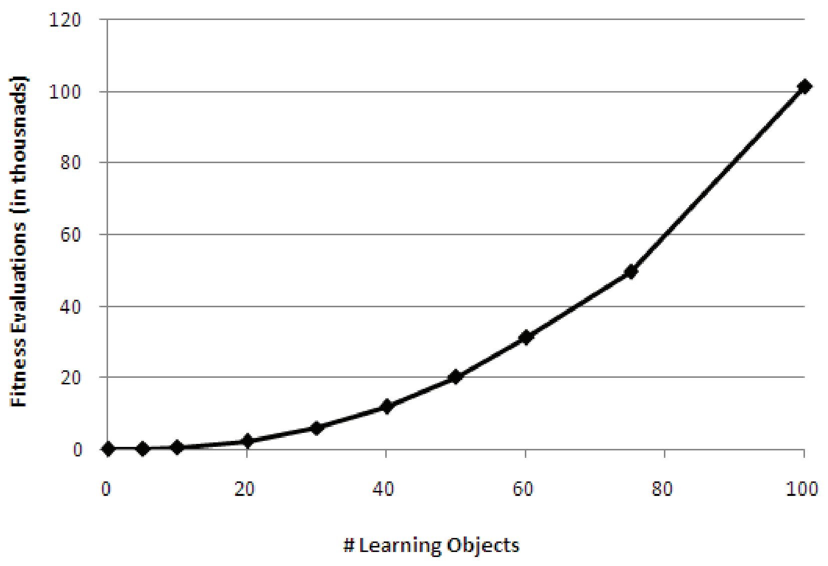

Fig. 6. Number of fitness evaluations required for different number of LOs

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 will be easily interchanged and processed by compliant systems. (2) Instructor's role is automated reducing costs. Sequencing process works even in complex scenarios where humans face difficulties. Instructors could spend saved time in performing other activities within the learning action. And (3), the model can be extended to an automated intelligent system for building personalized e-learning experiences. But this third implication is linked to future work. This model has been envisaged and it was depicted in figure 3 (Section 2.2). Sequencing process can be complemented with gap analysis process and competency learner modelling techniques to build personalized courses. These courses could also be SCORM (ADL, 2004) compliant, so they could be imported to current LMSs.

6. 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 program (grants FIT-350200-2007-6 and FIT-350101-2007-9) and Plan Avanza program (grant PAV-070000-2007-103), (3) the Spanish Ministry of Education and Science PROFIT program (grant CIT-410000-2007-5), (4) Castilla-La Mancha autonomous community under the educational innovation cooperation program (grant EM2007-004) by (5) research groups' support program from the University of Alcalá and CAM Madrid Region (grant CCG06-UAH/TIC-0732).

7. References

ADL (2004) Shareable Content Object Reference Model (SCORM). The SCORM 2004 Overview. Advanced Distributed Learning (ADL) Initiative.

Barchino, R.; Gutiérrez, J. M. & Otón, S. (2005) *An Example of Learning Management System*. In Isaías, P., Baptista, M. & Palma, A. (Eds.) *IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005).* Virtual, IADIS Press.

Barr, A. (2006) Revisiting the -ilities: Adjusting the Distributed Learning Marketplace, Again? *Learning Technology Newsletter,* 8**,** 3-4.

Brusilovsky, P. (1996) Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction,* 6**,** 87-129.

Brusilovsky, P. (1999) Adaptive and Intelligent Technologies for Web-based Education. *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching,* 4**,** 19-25.

CEN/ISSS (2006) European Model for Learner Competencies. Comité Européen de Normalisation / Information Society Standardization System (CEN/ISSS).

De Bra, P.; Aerts, A.; Berden, B.; Lange, B. D.; Rousseau, B.; Santic, T.; Smits, D. & Stash, N. (2003) AHA! The adaptive hypermedia architecture. *Proceedings of the fourteenth ACM conference on Hypertext and hypermedia.* Nottingham, UK, ACM Press.

De Bra, P., Houben, G.-J. & Wu, H. (1999) AHAM: a Dexter-based reference model for adaptive hypermedia. *Proceedings of the tenth ACM Conference on Hypertext and hypermedia.* Darmstadt, Germany, ACM Press.

Eberhart, R. & Kennedy, J. (1995) A new optimizer using particle swarm theory. *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS '95.* Nagoya, Japan.

Hinchey, M. G., Sterritt, R. & Rouff, C. (2007) Swarms and Swarm Intelligence. *Computer,* 40**,** 111-113.

HR-XML (2006) Competencies (Measurable Characteristics) Recommendation. HR-XML Consortium.

Hu, X., Eberhart, R. C. & Shi, Y. (2003) Swarm intelligence for permutation optimization: a case study of n-queens problem. *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* Indianapolis, USA, IEEE Press.

IEEE (2002) Learning Technology Standards Committee (LTSC). Learning Object Metadata (LOM). 1484.12.1. IEEE.

IEEE (2008) Learning Technology Standards Committee (LTSC). Standard for LearningTechnology - Data Model for Reusable Competency Definitions. IEEE.

IMS (2002a) Reusable Definition of Competency or Educational Objective - Best Practice and Implementation Guide. IMS Global Learning Consortium.

IMS (2002b) Reusable Definition of Competency or Educational Objective - Information Model. IMS Global Learning Consortium.

Karampiperis, P. (2006) Automatic Learning Object Selection and Sequencing in Web-Based Intelligent Learning Systems. IN ZONGMIN, M. (Ed.) *Web-Based Intelligent E-Learning Systems: Technologies and Applications.* London. UK., Idea Group.

Kennedy, J. & Eberhart, R. (1995) Particle swarm optimization. *Proceedings., IEEE International Conference on Neural Networks.* Perth, WA, Australia.

Kennedy, J. & Eberhart, R. C. (1997) A discrete binary version of the particle swarm algorithm. *1997 IEEE International Conference on Systems, Man, and Cybernetics. 'Computational Cybernetics and Simulation'.*

Mendes, R., Kennedy, J. & Neves, J. (2004) The fully informed particle swarm: simpler, maybe better. *Evolutionary Computation, IEEE Transactions on,* 8**,** 204-210.

Robinson, J. & Rahmat-Samii, Y. (2004) Particle swarm optimization in electromagnetics. *Antennas and Propagation, IEEE Transactions on,* 52**,** 397-407.

Schoofs, L. & Naudts, B. (2000) Ant colonies are good at solving constraint satisfaction problems. *Proceedings of the 2000 Congress on Evolutionary Computation.* La Jolla, CA.

Tsang, E. (1993) *Foundations of Constraint Satisfaction,* Academic Press.

Van Den Berg, B., Van Es, R., Tattersall, C., Janssen, J., Manderveld, J., Brouns, F., Kurvers, H. & Koper, R. (2005) Swarm-based sequencing recommendations in e-learning. *Proceedings 5th International Conference on Intelligent Systems Design and Applications, 2005. ISDA '05.* Wroclaw, Poland.

Wiley, D. A. (2000) Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. IN WILEY, D. A. (Ed.) *The Instructional Use of Learning Objects.*

Wilkinson, J. (2001) A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM. *IEEE International Conference on Advanced Learning Technologies, 2001. Proceedings.*