Swarm Intelligence in e-Learning: A Learning Object Sequencing Agent based on Competencies

Luis de-Marcos
Computer Science Department
Polytechnic School
University of Alcala. Spain
+34 91 885 66 51
luis.demarcos@uah.es

José-Javier Martínez
Computer Science Department
Polytechnic School
University of Alcala. Spain
+34 91 885 66 51
josej.martinez@uah.es

José-Antonio Gutiérrez
Computer Science Department
Polytechnic School
University of Alcala. Spain
+34 91 885 66 54
jantonio.gutierrez@uah.es

ABSTRACT
In e-learning initiatives content creators are usually required to arrange a set of learning resources in order to present them in a comprehensive way to the learner. Course materials are usually divided into reusable chunks called Learning Objects (LOs) and the ordered set of LOs is called sequence, so the process is called LO sequencing. In this paper an intelligent agent that performs the LO sequencing process is presented. Metadata and competencies are used to define relations between LOs so that the sequencing problem can be characterized as a Constraint Satisfaction Problem (CSP) and artificial intelligent techniques can be used to solve it. A Particle Swarm Optimization (PSO) agent is proposed, built, tuned and tested. Results show that the agent succeeds in solving the problem and that it handles reasonably combinatorial explosion inherent to this kind of problems.

Categories and Subject Descriptors
K.3.1 [Computers and Education] Computer Uses in Education - *Computer-assisted instruction (CAI)*.

General Terms
Experimentation, Standardization.

Keywords
e-Learning, Learning Object (LO), Learning Object Sequencing, Competency, Particle Swarm Optimization (PSO), Swarm Intelligence

1. INTRODUCTION
Brusilovsky [5] 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 (and so be used in) to 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 adaptive hypermedia field [6] and rely on complex conceptual models, usually driven by sequencing rules [9, 20]. E-learning traditional approaches and paradigms, that promote reusability and interoperability, are generally ignored, thus resulting in (adaptive) proprietary systems (such as AHA! [8]) and non-portable courseware. But e-learning approaches also expose its own problems. 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" [27]. But offering these 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" [4] and, among these roles, sequencing seems to be a priority.

In this paper an innovative sequencing technique that automates teacher´s role is proposed. E-Learning standards and learning object paradigm are used 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 modeled 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 shown 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) as well as in simulated scenarios. 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) [28].

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.

LOs must be arranged in a suitable sequence previously 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. Most commonly used specification is SCORM [2]. Courseware that conforms SCORM´s Content Aggregation Model [1] 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 Definitions

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" [29]. This definition emphasizes that competencies comprise not only knowledge, but they also embrace a set of factors which 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 [18] describes a competency as four-dimensional element (figure. 1).

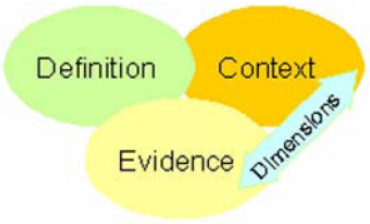

Figure 1. RDCEO competency conceptual model (from [18])

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 [19],
- IEEE Learning Technology Standards Committee (LTSC) "Draft Standard for Learning Technology - Standard for Reusable Competency Definitions " specification (currently an approved draft) [15],
- HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" [14].
- CEN/ISSS "A European Model for Learner Competencies" workshop agreement [7]

All these specifications offer 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 pratitioners with a huge set of entities, fields and relations that they must fulfill 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.

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 (figure 2), a constraint between the two LOs is established so that the first LO must precede the second one in a valid sequence.

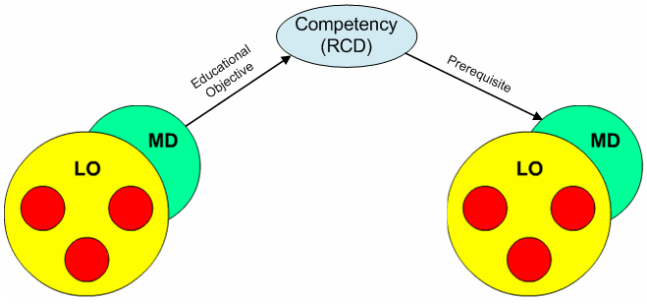

Figure 2. LO sequencing through competencies

Meta-Data (MD) definitions are attached to LOs, and within those definitions references to competencies (prerequisites and learning outcomes) are included. LOM [16] records have been used for specifying LO Meta-Data. LOM element 9, 'Classification', is used to include competency references as recommended in [17,

18]. 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.

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 modeling prerequisites and learning outcomes, because competencies are also useful for modeling user current knowledge and learning initiatives' expected outcomes (future learner knowledge).We are proposing a wider framework (figure 3) in which learner (user) modeling 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.

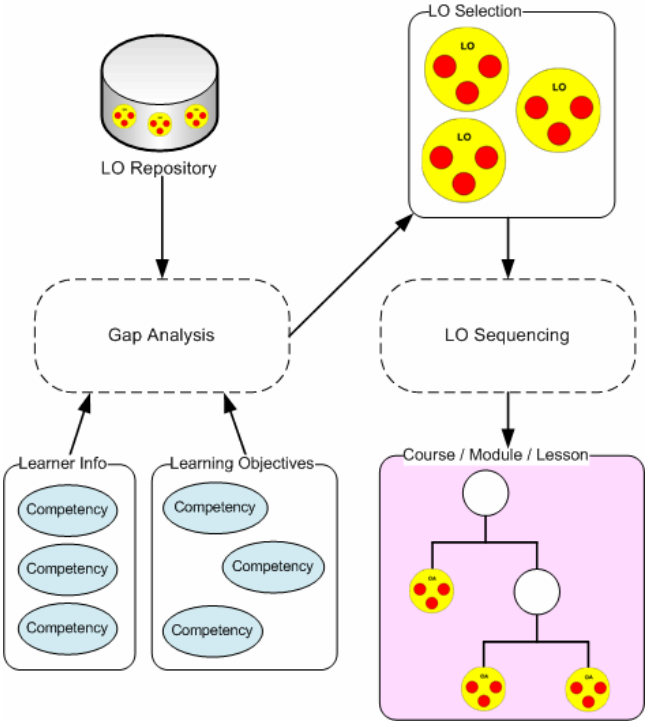

Figure 3. Competency-driven content generation model

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. 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 [26] a CSP is triple *(X,D,C)* where

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

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 adjust dynamically 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. PSO has been used to solve a wide variety of problems [13].

Original PSO [11, 21] is intended to work on continuous spaces, and velocity is computed for each dimension. 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 [22]. 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 [30]. 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, but it is normalized to the range 0 to 1 by dividing each element by the maximum value of the particle (i.e. of all elements). Mutation is also introduced; 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 [30] 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 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 research in an area that has been called sociometry. Topologies define structures that determine neighborhood relations, and several of them (ring, cluster, pyramid, square and all topologies) have been studied. It has been demonstrated that fully informed approaches outperform all other methods [23]. The fully informed approach prompts using '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. Table1 includes LO sequencing agent pseudo code. Several issues concerning design and implementation were decided. In the rest of this section each of these issues is discussed and the selection criteria are explained.

3.3.1 Fitness Function

It is critical to choose a function that accurately represents the goodness of a solution [24]. 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 [25]:

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

where $V_{i,j} : D_i \times D_j \rightarrow \{0,1\}$ is the violation function

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

The standard penalty function returns the number of constraints violated, so PSO objective is to minimize that function. 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 $C= \{x_i-x_j > 0 : x_i \in X, x_j \in \{x_1,\ldots,x_i\}\}$. 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 it is used the second definition. 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, 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 process 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 (user obligations are lightened)

3.3.2 PSO Parameters

One important PSO advantage is that it uses a relative small number of parameters compared with other techniques like genetic algorithms. However, much literature on PSO parameter subject has been written. Among it, Xiaohui et. al in [30] 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 following equation:

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

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.

3.3.3 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.

3.3.4 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 number of iterations setting can avoid infinite computing.

3.3.5 Tuning.

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, three different tuning ways were envisaged in order to improve agent

performance. First option is to change *pbest* and *gbest* values when an equal or best fitness value is 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 (<=). Original algorithm determines that *pbest* and *gbest* only change if a better state is found (comparisons <). Second tuning mechanism is to randomly decide whether the permutation of a particle's position was performed from *gbest* or from *pbest* (p=0.5). In the original version all permutations are done regarding *gbest*. These changes resemble to be quite logical ways for increasing particles' mobility and for avoiding quick convergence to local minimums.

Finally, when the implementation was finished and test suites were being launched a deeper knowledge of the solution space was acquired and an additional mechanism was introduced due to the following fact: It could be observed that in huge solution spaces some velocity values tend to grow indefinably and fast in one direction. So that these 'great' values reduce the probability assigned to other values from moving towards *gbest* when normalized velocity is computed. This problem was avoided introducing a special function that limits the velocity of each value to a maximum value. In our opinion, this value must not be a fixed parameter and that it must depend on the number of learning objects that comprise the sequence. Initially, it was decided to set the velocity limit equal to the number of LOs in the sequence. Therefore, each velocity value of the normalized velocity vector (called V_{norm}) is not allowed to grow beyond a maximum value equal to the number of learning objects in the sequence. This setting also intends to introduce a massive movement towards *gbest* when the number of iterations increase and all the velocity values reach that limit, so that the region close to *gbest* is explored. It should be noted that mutation ensures that these particles are close to but not equal to *gbest* in order to not lose computational resources exploring the same solution repeatedly. The final agent pseudo-code is presented in table 1.

Table 1. PSO sequencing agent pseudo-code

```
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
      Vnew = w × Vold+ (c1×rnd()×(pbest-currentValue)) +
             (c2×rnd()×(gbest-currentValue))
    Normalize Velocity as
      Vnorm = Vnew/max(Vnew)
    Check Vnorm limit
      for each v in Vnorm {
        if(v > length(X))
          v = length(X)
      }
    Update particle value
      for i = 1 to length(Vnorm) {
        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
```

where *currentValue* is a vector of *n* learning objects representing the current position of the particle (state or solution being computed), and, V_{new}, V_{old} and V_{norm} are vectors of *n* positions representing different velocities required by the algorithm.

4. EXPERIMENTAL RESULTS

The PSO algorithm for LOs sequencing described above was designed and implemented using the object oriented paradigm. We wanted to test its performance in real and simulated scenarios. As a real-world problem, we choose a problem concerning course sequencing for a Master in Engineering (M.Eng.) program in our institution. 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 [3], 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 estimation 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 first two tuning mechanism 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 o 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 (<).

Each configuration was run 100 times and results representing mean fitness values' evolution were computed (figure 4). All configurations converge to a feasible solution, but configuration 1

(original settings) outperform all others. Configurations 1 and 2 show similar performance but configuration 1 reaches before any other a 100% success ratio for 100 runs.

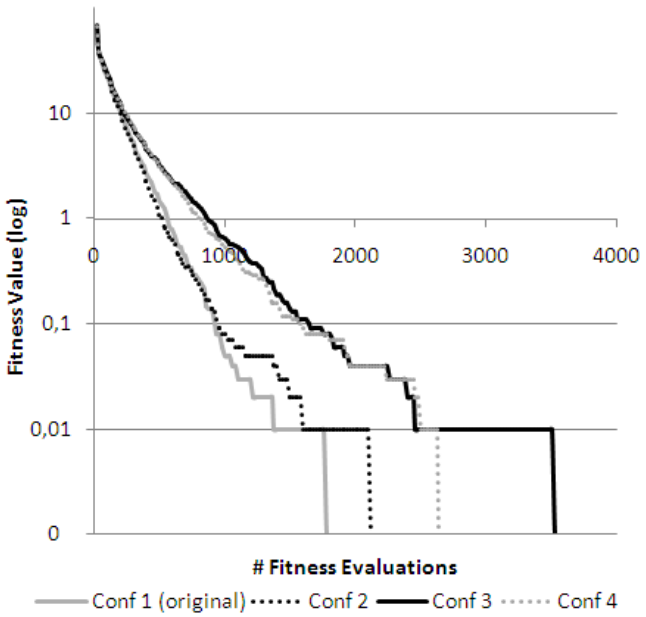

Figure 4. PSO Configurations performance comparison

All these tests were run checking the normalized velocity limit (third proposed tuning mechanism). In order to test its the real performance, the four configuration sets where run without performing the velocity check. Table 2 compares the results obtained in both cases by showing the mean values required for 100 runs to reach a solution. As it can be shown velocity check dramatically improves performance. Furthermore, original settings (concerning the other two tuning mechanism) also displays better performance for both cases.

Table 2. Mean number of fitness evaluations for each configuration with and without normalized velocity check

Configuration	μ Fitness evaluations without velocity check	μ Fitness evaluations with velocity check
Configuration 1	1158	641
Configuration 2	1237	645
Configuration 3	1817	1008
Configuration 4	1412	975

The tested scenario may seem to have many feasible solutions that would make doubtful PSO performance in more 'challenging' scenarios, so additional test were conducted. Test sequences of 5, 10, 20, 30, 40, 50, 60, 75 and 100 LOs with only one feasible solution were designed. Each test suite was run 100 times with and without the velocity check and mean values were computed. Best configuration concerning the other tuning mechanism was set. Figure 5 shows the results and it supports the argument that velocity control improves agent performance as the solution space size grows. It could also be inferred that the proposed PSO agent handles reasonably combinatorial explosion for this particular problem. It should be noted that while the number of learning objects grows linearly the size of the solution space grows exponentially.

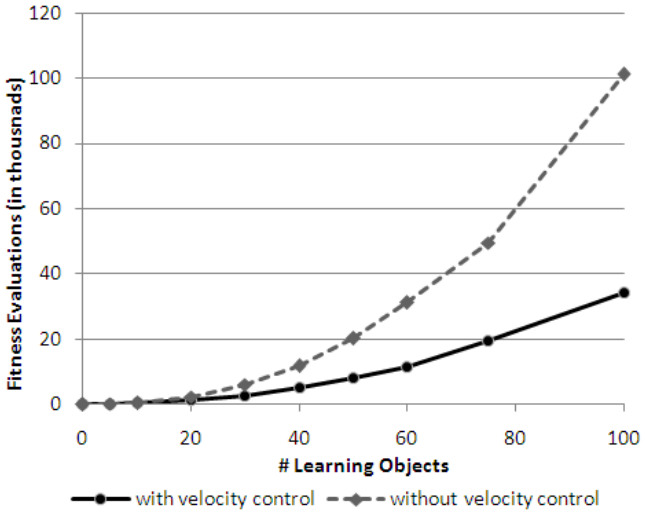

Figure 5. Number of fitness evaluations required for different number of LOs

5. CONCLUSIONS

Automated LO sequencing is a recurring problem in the e-learning field that could be approached employing models that ensure interoperability along with 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 represented 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 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 permutation problem has been extended to LO sequencing problem, and it has also been properly tuned. Results show that: (1) PSO succeeds in solving the problem, (2) the original configuration is the best one, and (3) a velocity check that limits the normalized velocity of each particle value improves performance in the tested scenarios.

Further implications arise from the model proposal (section 2): (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 were 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 modeling techniques to build personalized courses. These courses could also be SCORM [2] compliant, so they could be imported to current LMSs.

Finally, other AI techniques should be analyzed to test its performance for solving the LO sequencing problem. Particularly, Ant Colony Optimization (ACO) [10] and genetic algorithms have demonstrated optimal results regarding CSP solving [12]. Exact techniques may also be considered. We plan to design and build intelligent sequencing agents using these techniques and check its results against PSO implementation performance.

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) and (4) Castilla-La Mancha autonomous community under the educational innovation cooperation program (grant EM2007-004). Authors also want to acknowledge support from the TIFyC research group.

7. REFERENCES

[1] ADL. Shareable Content Object Reference Model (SCORM). The SCORM 2004 Content Aggregation Model, Advanced Distributed Learning (ADL) Initiative, 2004.

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

[3] Barchino, R., Gutiérrez, J.M. and Otón, S., An Example of Learning Management System. in IADIS Virtual Multi Conference on Computer Science and Information Systems (MCCSIS 2005), (Virtual, 2005), IADIS Press, 140-141.

[4] Barr, A. Revisiting the -ilities: Adjusting the Distributed Learning Marketplace, Again ? Learning Technology Newsletter, 8 (1/2). 3-4.

[5] Brusilovsky, P. Adaptive and Intelligent Technologies for Web-based Education. Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching, 4. 19-25.

[6] Brusilovsky, P. Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction, 6 (2 - 3). 87-129.

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

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

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

[10] Dorigo, M. and Caro, G.D. The Ant Colony Optimization meta-heuristic. in Corne, D., Dorigo, M. and Glover, F. eds. New Ideas in Optimization, McGraw Hill, London, UK, 1999, 11-32.

[11] Eberhart, R. and Kennedy, J., 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), 39-43.

[12] Eiben, A.E. and Smith, J.E. Introduction to Evolutionary Computing. Springer-Verlag, Berlin (Germany), 2003.

[13] Hinchey, M.G., Sterritt, R. and Rouff, C. Swarms and Swarm Intelligence. Computer, 40 (4). 111-113.

[14] HR-XML. Competencies (Measurable Characteristics) Recommendation, HR-XML Consortium, 2006.

[15] IEEE. Learning Technology Standards Committee (LTSC). Draft Standard for LearningTechnology - Data Model for Reusable Competency Definitions, IEEE, 2007.

[16] IEEE. Learning Technology Standards Committee (LTSC). Learning Object Metadata (LOM). 1484.12.1, IEEE, 2002.

[17] 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.

[18] IMS. Reusable Definition of Competency or Educational Objective - Best Practice and Implementation Guide, IMS Global Learning Consortium, 2002.

[19] IMS. Reusable Definition of Competency or Educational Objective - Information Model, IMS Global Learning Consortium, 2002.

[20] Karampiperis, P. 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, Idea Group, London. UK., 2006.

[21] Kennedy, J. and Eberhart, R., Particle swarm optimization. in Proceedings., IEEE International Conference on Neural Networks., (Perth, WA, Australia, 1995), 1942-1948 vol.1944.

[22] Kennedy, J. and Eberhart, R.C., A discrete binary version of the particle swarm algorithm. in 1997 IEEE International Conference on Systems, Man, and Cybernetics. 'Computational Cybernetics and Simulation'. (1997), 4104-4108.

[23] Mendes, R., Kennedy, J. and Neves, J. The fully informed particle swarm: simpler, maybe better. Evolutionary Computation, IEEE Transactions on, 8 (3). 204-210.

[24] Robinson, J. and Rahmat-Samii, Y. Particle swarm optimization in electromagnetics. Antennas and Propagation, IEEE Transactions on, 52 (2). 397-407.

[25] Schoofs, L. and Naudts, B., Ant colonies are good at solving constraint satisfaction problems. in Proceedings of the 2000 Congress on Evolutionary Computation., (La Jolla, CA, 2000), 1190-1195.

[26] Tsang, E. Foundations of Constraint Satisfaction, Academic Press, 1993.

[27] van den Berg, B., van Es, R., Tattersall, C., Janssen, J., Manderveld, J., Brouns, F., Kurvers, H. and Koper, R., Swarm-based sequencing recommendations in e-learning. in Proceedings 5th International Conference on Intelligent Systems Design and Applications, 2005. ISDA '05., (Wroclaw, Poland, 2005), 488-493.

[28] Wiley, D.A. 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, 2000.

[29] Wilkinson, J., 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), 205-208.

[30] Xiaohui, H., Eberhart, R.C. and Yuhui, S., Swarm intelligence for permutation optimization: a case study of n-queens problem. in Proceedings of the 2003 IEEE Swarm Intelligence Symposium, (Indianapolis, USA, 2003), IEEE Press, 243-246.