Competency-based Learning Object Sequencing using Particle Swarms

L. de-Marcos, C. Pages, J.J. Martínez, J.A. Gutiérrez
Computer Science Department. University of Alcalá.
Ctra. Barcelona km 33.6, Alcalá de Henares, Spain
{luis.demarcos, carmina.pages, josej.martinez, jantonio.gutierrez}@uah.es

Abstract

In e-learning initiatives, sequencing problem concerns arranging a particular set of learning units in a suitable succession for a particular learner. Sequencing is usually performed by instructors, who create general and 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 promoted 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 AI techniques can be used to solve it. Particle Swarm Optimization (PSO) is one of such techniques and it has proven with good performance solving a wide variety of problems. An implementation of the PSO, for learning object sequencing, is presented and its performance in a real scenario is discussed.

1. Introduction

Brusilovsky [1] envisages 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 students with the most suitable individually planned sequence of knowledge units to learn and sequence the learning tasks … to work with". These methods derive from adaptive hypermedia field [2] and rely on complex conceptual models, usually driven by sequencing rules [3, 4]. E-learning traditional approaches and paradigms, that promote reusability and interoperability, are generally ignored, thus resulting in (adaptive) proprietary systems (such as AHA! [5]) and non-portable courseware.

In this paper an innovative sequencing technique 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 Particle Swarm Optimization (PSO) is used to find a suitable sequence within the solution space respecting the constraints. In section 2, the problem model for competency-based learning object sequencing is presented. Section 3 describes the particle swarm optimization approach for solving the problem. Section 4 presents the results obtained from implementing and testing the intelligent algorithm in a real world situation (course sequencing in an online Master in Engineering program). And finally Section 5 depicts conclusions and future research lines.

2. Learning Objects and 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) [6].

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 create generic courses, targeting generic learner profiles. These sequences are then coded using a standard specification to ensure interoperability. Most commonly used specification is SCORM [7]. Courseware that conforms SCORM´s Content Aggregation Model [8] is virtually portable between a wide variety of Learning Management Systems (LMSs). Though, SCORM usage hinders the automatic LO sequencing due to its system-centered vision. Other metadata-driven approaches offer better possibilities. Just LO metadata will enable automatic sequencing process to be performed. And the appropriate combination of metadata and competencies will enable adaptive and automatic content sequencing.

2.1. Competencies for interoperable Learning Object Sequencing

Competencies can be formally described as "multidimensional, comprised of knowledge, skills and psychological factors that are brought together in complex behavioral responses to environmental cues" [9]. Some e-learning trends are trying to standardize competency definitions so that they could be interchanged and processed by machines. It is worth quoting the following specifications:

- IMS "Reusable Definition of Competency or Educational Objective" (RDCEO) specification [10],
- IEEE Learning Technology Standards Committee (LTSC) "Draft Standard for Learning Technology - Standard for Reusable Competency Definitions " specification (currently an approved draft) [11],
- and HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" [12].

According to RDCEO and IEEE nomenclature, a competency record is called 'Reusable Competency Definition' (RCD). RCDs can be attached to LOs in order to define their prerequisites and their 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 defining the same competency as the prerequisite for another LO (fig 1), a constraint between the two LOs is established so that the first one must precede the second LO in a valid sequence.

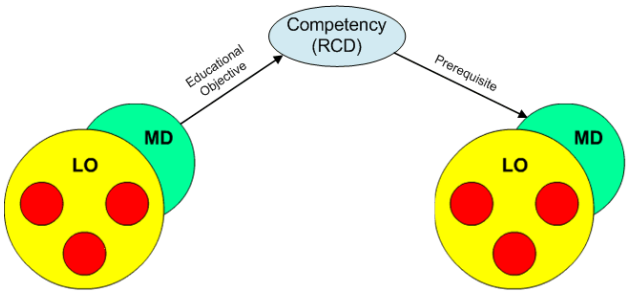

Fig 1. LO sequencing through competencies

Metadata (MD) definitions are attached to LOs, and within those definitions references to competencies (prerequisites and learning outcomes) are included. LOM [13] records have been used for specifying LO metadata. LOM element 9, 'Classification', is used to include competency references as recommended in [14, 15]. 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. So, 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 (see Section 5) in which learner (user) modeling is done in terms of competencies, and these competencies are also used to define the expected learning outcomes from a learning program.

3. Particle Swarm Optimization in Learning Object Sequencing

Given a random LOs' sequence modeled as described above, the question of finding a correct sequence can be envisaged as a classical artificial intelligent Constraint Satisfaction Problem (CSP). In this manner, 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 between states. So we face a permutation problem, which is a special kind of CSP.

Particle Swarm Optimization (PSO) is an evolutionary computing optimization algorithm. PSO mimics the behavior 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 their velocity and cooperate towards finding a solution. Best solution found: (1) by a particle is called *pbest*, (2) within a set of neighbor particles is called *nbest*, (3) and within the whole swarm is called *gbest*. PSO have been used to solve a wide variety of problems [16].

Original PSO [17, 18] is intended to work on continuous spaces. A discrete binary version was presented in [19]. 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 mutation concept was also introduced in the permutPSO 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] is 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.

Figure 2 presents the basic PSO procedure for LO sequencing.

```
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
      V_new = w × V_old + c1 × rand() × (P_pbest − X)
              + c2 × rand() × (P_nbest − 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 currentValue[i] for
            indexOf(currentValue, gBest[i])
      }
    Check Mutation
      if (currentValue = gBest) swap two
        random positions from currentValue
  }
} until termination criterion is met
```

Fig 2. PSO Procedure for LO Sequencing

For an adequate PSO implementation fitness function and parameters should also be carefully selected.

Fitness Function. It is critical to choose a function that accurately represents the goodness of a solution [21]. The following function is suggested:

$$\text{fitness}(s) = \sum_{i=0}^{n} s[i].\text{pr}_n \quad (1)$$

Where *s* is the LO sequence, *n* is the number of LOs in *s*, *s[i]* is the i-th LO in the sequence, and pr_n is the number of prerequisites in a LO not delivered by their predecessors in the sequence. pr_n is computed using a function that recursively process all outcomes delivered by previous LOs in the sequence, checking for each prerequisite accomplishment. The fitness value of a feasible solution should be zero, so PSO tries to minimize this function. When a solution fitness function call returns 0, the operation of the algorithm is stopped returning the current state (solution).

PSO Parameters. One important advantage of PSO 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. [20] established the set of parameters so that PSO works properly for solving permutation problems. So we decided to take 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:

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

Population size was set to 20 particles and the fully informed version of PSO was used. 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

Proposed improvements. During the initial agent development we find 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 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 improvement 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*. Figure 3 presents these improvements. Changes respecting the basic procedure are showed in boldface.

```
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
      V_new = w × V_old + c1 × rand() × (P_pbest − X)
              + c2 × rand() × (P_nbest − 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 currentValue[i] for
              indexOf(currentValue, pBest[i])
          else
            swap currentValue[i] for
              indexOf(currentValue, gBest[i])
      }
    Check Mutation
      if (currentValue = gBest) swap two
        random positions from currentValue
  }
} until termination criterion is met
```

Fig 3. Improvements on PSO Procedure

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

4. Results

The PSO algorithm for LOs sequencing described above was implemented using Microsoft Visual Studio C#. 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 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 (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 regarding any other course may be set.

All courses have a (expected) learning time that range from 30 to 50 hours. They are delivered online using a LMS and they have their metadata records. Competency records were created to specify LOs' restrictions, and LOs' 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. Just estimations have been used. We have estimated that the relation between feasible solutions and total solutions order is $8{,}9 \times 10^{12}$. 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 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 (<).

Figure 4 shows the results for the four configurations. Each configuration was run 1000 times and the results represent the succeed ratio. From the results, it can be seen that all configurations converge to a feasible solution, but configuration 4 (original settings) outperform all others.

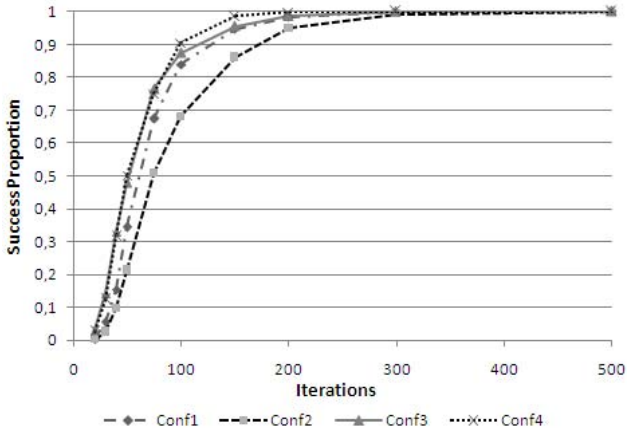

Fig 4. PSO Configurations Comparison

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).

Table 1. Mean number of fitness evaluations

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

The tested scenario may seem to have many feasible solutions that would make doubtful PSO performance in challenging scenarios, so PSO agent was tested in 'more difficult' situations. Test sequences of 20 (10^{18} solutions) and 26 (10^{36} solutions) LOs, where there was only one feasible solution, were designed. The agent succeeded in finding the solution if 200 and 500 iterations were set, respectively, in each test (100% success in 1000 runs).

5. Conclusions and Future Work

The purpose of the study was to design, develop and test a PSO agent that performs automatic LO sequencing through competencies. The PSO for permutation problem have 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.

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, with reduced costs. Sequencing process works even in complex scenarios were humans face difficulties. And (3), the model can be extended to an automated intelligent system for building personalized e-learning experiences. But this third implication is more appertained to future work. This model has been envisaged and is depicted in figure 5. Sequencing process can be complemented with gap analysis process and competency learner modeling techniques to build personalized courses. This courses could also be SCORM [7] compliant, so they could be imported to current LMSs.

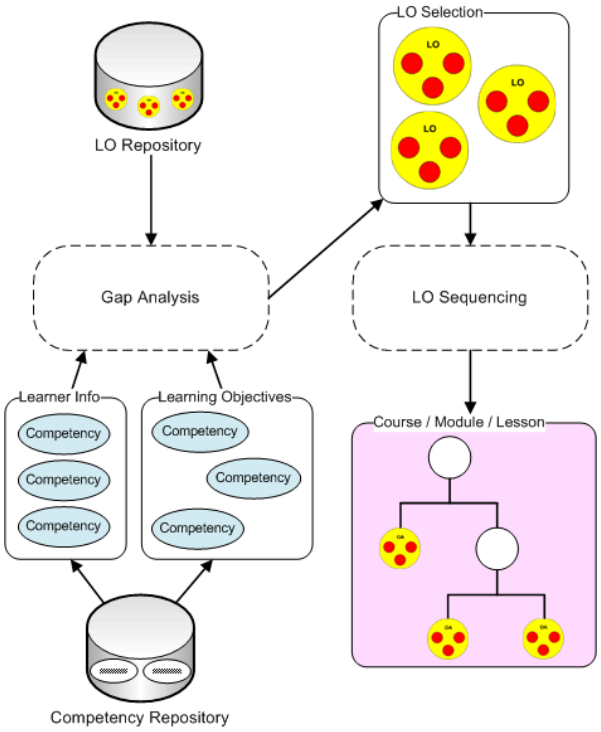

Fig 5. Competency-driven content generation model

Finally, other intelligent sequencing techniques should be analyzed. Particularly, ant colony optimization (ACO) [22, 23] and genetic algorithms have proven optimal results regarding CSP solving. 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 FPI research staff education program of the University of Alcalá, (2) the Spanish Ministry of Industry PROFIT program (SNOA project under contract FIT-350200-2007-6), and by (3) research groups' support program from the University of Alcalá (grant CCG06-UAH/TIC-0732). Authors also want to acknowledge support from the TIFyC research group.

7. References

[1] P. Brusilovsky, "Adaptive and Intelligent Technologies for Web-based Education," *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching,* vol. 4, pp. 19-25, 1999.

[2] P. Brusilovsky, "Methods and techniques of adaptive hypermedia," *User Modeling and User-Adapted Interaction,* vol. 6, pp. 87-129, 1996.

[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 LO Selection and Sequencing in Web-Based Intelligent Learning Systems," in *Web-Based Intelligent E-Learning Systems: Technologies and Applications*, M. Zongmin, Ed. London. UK.: Idea Group, 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* Nottingham, UK: ACM Press, 2003.

[6] D. A. Wiley, "Connecting LOs to instructional design theory: A definition, a metaphor, and a taxonomy," in *The Instructional Use of LOs*, D. A. Wiley, Ed., 2000.

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

[8] ADL, "Shareable Content Object Reference Model (SCORM). The SCORM 2004 Content Aggregation Model." vol. 2005: Advanced Distributed Learning (ADL) Initiative, 2004.

[9] 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.

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

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

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

[13] IEEE, "Learning Technology Standards Committee (LTSC). LO Metadata (LOM). 1484.12.1," IEEE, 2002.

[14] IEEE, "Learning Technology Standards Committee (LTSC). Standard for Learning Technology—Extensible Markup Language (XML) Schema Definition Language Binding for LO Metadata. 1484.12.3.," IEEE, 2005.
[15] IMS, "Reusable Definition of Competency or Educational Objective - Best Practice and Implementation Guide," IMS Global Learning Consortium, 2002.
[16] M. G. Hinchey, R. Sterritt, and C. Rouff, "Swarms and Swarm Intelligence," *Computer,* vol. 40, pp. 111-113, 2007.
[17] 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.
[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings., IEEE International Conference on Neural Networks.*, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4.
[19] 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 vol.5.
[20] H. Xiaohui, R. C. Eberhart, and S. Yuhui, "Swarm intelligence for permutation optimization: a case study of n-queens problem," in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003. SIS '03.* , 2003, pp. 243-246.
[21] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *Antennas and Propagation, IEEE Transactions on,* vol. 52, pp. 397-407, 2004.
[22] M. Dorigo and G. D. Caro, "The Ant Colony Optimization meta-heuristic," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glover, Eds. London, UK: McGraw Hill, 1999, pp. 11-32.
[23] M. Dorigo, G. D. Caro, and L. M. Gambardella, "Ant algorithms for discrete optimization," *Artificial Life,* vol. 5, pp. 137-172, 1999.