Particle Swarms for Competency-based Curriculum Sequencing

Luis de-Marcos[1] , José-Javier Martínez[1], José-Antonio Gutiérrez[1]

[1] Computer Science Department. University of Alcalá.
Ctra. Barcelona km 33.6. Alcalá de Henares, Spain
{luis.demarcos, josej.martinez, 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.

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

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 all 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. Current literature is surveyed and several enhancements over the original algorithm are proposed. 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 commercial 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 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” [8]. 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 efforts: (1) IMS "Reusable Definition of Competency or Educational Objective” (RDCEO) specification [9]. (2) IEEE Learning Technology Standards Committee (LTSC) “Standard for Learning Technology - Data Model for Reusable Competency Definitions" specification (currently an approved standard, pending publishing) [10]. (3) HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" [11]. And, CEN/ISSS “A European Model for Learner Competencies” workshop agreement [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. 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).

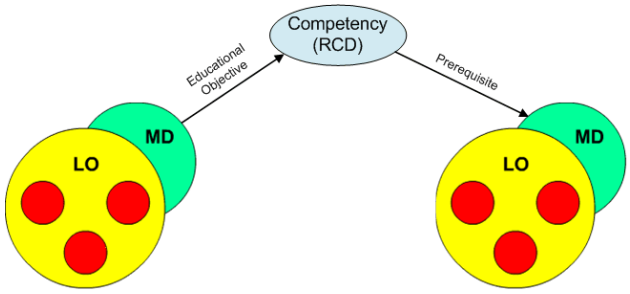

Fig 1. LO sequencing through competencies

3 Competency-based Intelligent Sequencing

Given a random LOs' sequence modeled as described above, the question of finding a correct sequence can be envisaged as a classical 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.

3.1 Mathematical Characterization

According to [16] a CSP is triple (X,D,C) where $X=\{x_0,x_1,\ldots,x_{(n-1)}\}$ is a finite set of variables, D is a function that maps each variable to its corresponding domain $D(X)$, and $C_{ij} \subset D_i \times D_j$ 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 then (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 $P=\langle v_0, v_1, \ldots, v_{n-1}\rangle$ is a tuple of $|X|=n$ values.

A solution *S* of a PermutCSP must be a solution of *(X,D,C)* and a complete permutation of *P*.

The learning object sequencing problem could be 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:

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

3.2 Particle Swarm Optimization for Learning Object Sequencing

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 [17].

Original PSO [18, 19] is intended to work on continuous spaces. A discrete binary version was presented in [20]. 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 [21]. 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 [21] 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. Discrete full-informed version [22] of the PSO was implemented in order to test its performance for solving the LO sequencing problem. But several other issues concerning design and implementation of the PSO were 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 [23]. A standard penalty function is a common choice for CSPs. We propose the following formula:

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

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, Hu et. al. [21] established the set of parameters so that PSO works properly for solving permutation problems. 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:

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

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 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 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*. 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 by the authors and an additional improvement 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. It seems evident that 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 (V_{norm}) is not allowed to grow beyond a maximum value equal to the number of learning objects in the sequence. This improvement 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 following code presents the final algorithm code with all these improvements.

```
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[i] in  Vnorm {
        if(v[i] > length(X))
          v[i] = length(X)
      }
    Update particle value
      for each v[i] in Vnorm {
          if(rand() < 0.5)
            swap currentValue[i] for
              currentValue[indexOf(pBest, currentValue[i]
          else
            swap currentValue[i] for
              currentValue[indexOf(gBest, currentValue[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 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). All of them must be completed before taking 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 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 [24] 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 $8{,}9x10^{12}$. 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 the first two proposed improvements introduced in Section 3. These configurations are:

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

Figure 2 shows the results for the four configurations. Each configuration was run 100 times and the results represent the mean fitness value evolution. From the results, it can be seen that 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 in 100 runs.

All these tests were run checking the normalized velocity limit (third proposed improvement in Section 3.2). In order to test the real performance of this improvement, the four configuration sets where run without performing the velocity check. Table 1 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 and original settings (concerning the other two improvements) also displays better performance for both cases.

The tested scenario may seem to have many feasible solutions that would make doubtful PSO performance in more 'challenging' scenarios, so PSO agent was tested in 'more difficult' situations. 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. Figure 3 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.

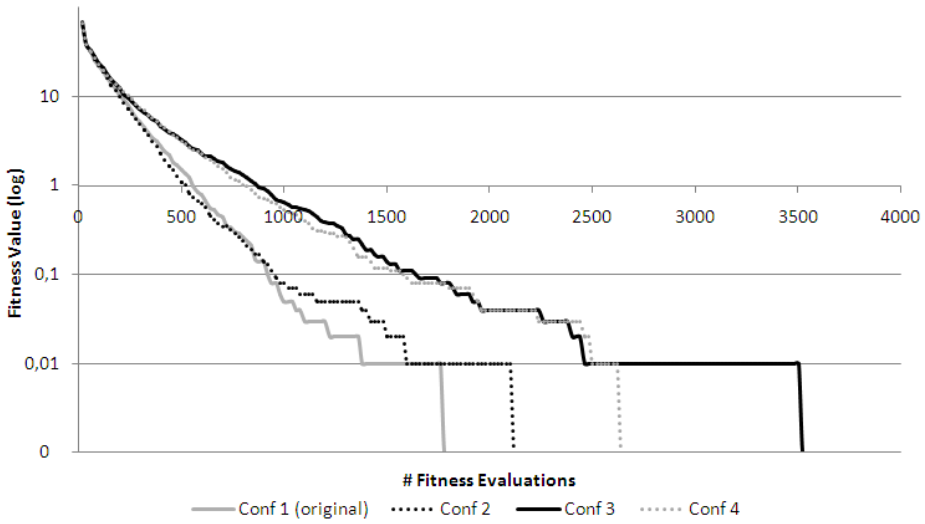

Fig 2. PSO Configurations performance comparison

Table 1. 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
Conf 1. comp <, permut gbest (original)	1158	641
Conf 2. comp <=, permut gbest	1237	645
Conf 3. comp <, permut gbest/pbest	1817	1008
Conf 4. comp <=, permut gbest/pbest	1412	975

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 for the LO sequencing problem. Testing three envisaged improvements was also performed. Results show that: (1) PSO succeeds in solving the problem, (2) the original configuration is the best one, and (3) velocity check for limiting the normalized velocity of each particle value improves performance in the tested scenarios.

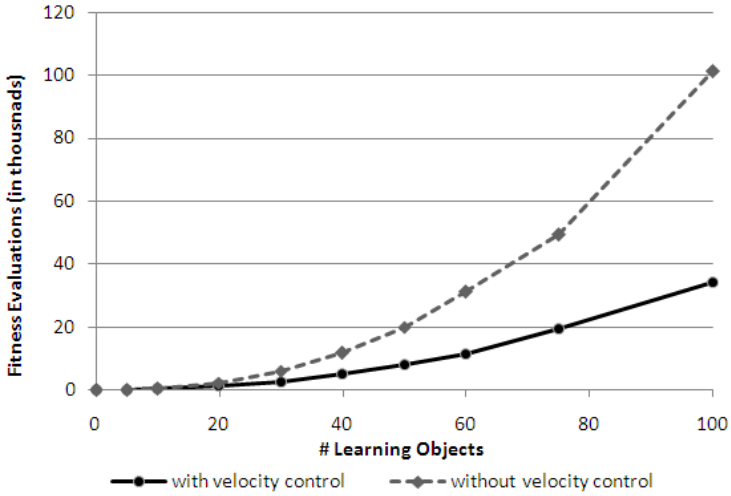

Fig 3. Number of fitness evaluations required for different number of LOs

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. 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. 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.

Acknowledgements. This research is co-funded by the University of Alcalá FPI research staff education program and by the Spanish Ministry of Industry, Tourism and Commerce Plan Avanza program (grant PAV-070000-2007-103)

References

1. Brusilovsky, P.: Adaptive and Intelligent Technologies for Web-based Education. Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching 4 (1999) 19-25
2. Brusilovsky, P.: Methods and techniques of adaptive hypermedia. User Modeling and User-Adapted Interaction 6 (1996) 87-129
3. De Bra, P., Houben, G.-J., 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)
4. 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)

5. De Bra, P., Aerts, A., Berden, B., Lange, B.d., Rousseau, B., Santic, T., Smits, D., Stash, N.: AHA! The adaptive hypermedia architecture. Proceedings of the fourteenth ACM conference on Hypertext and hypermedia. ACM Press, Nottingham, UK (2003)
6. 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)
7. ADL: Shareable Content Object Reference Model (SCORM). The SCORM 2004 Overview. Advanced Distributed Learning (ADL) Initiative (2004)
8. Wilkinson, J.: 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 (2001) 205-208
9. IMS: Reusable Definition of Competency or Educational Objective - Information Model. IMS Global Learning Consortium (2002)
10. IEEE: Learning Technology Standards Committee (LTSC). Draft Standard for LearningTechnology - Data Model for Reusable Competency Definitions. IEEE (2007)
11. HR-XML: Competencies (Measurable Characteristics) Recommendation. HR-XML Consortium (2006)
12. CEN/ISSS: European Model for Learner Competencies. Comité Européen de Normalisation / Information Society Standardization System (CEN/ISSS) (2006)
13. IEEE: Learning Technology Standards Committee (LTSC). Learning Object 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 Learning Object 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. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1993)
17. Hinchey, M.G., Sterritt, R., Rouff, C.: Swarms and Swarm Intelligence. Computer 40 (2007) 111-113
18. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS '95., Nagoya, Japan (1995) 39-43
19. Kennedy, J., Eberhart, R.: Particle swarm optimization. Proceedings., IEEE International Conference on Neural Networks., Vol. 4, Perth, WA, Australia (1995) 1942-1948 vol.1944
20. Kennedy, J., Eberhart, R.C.: A discrete binary version of the particle swarm algorithm. 1997 IEEE International Conference on Systems, Man, and Cybernetics. 'Computational Cybernetics and Simulation'. Vol. 5 (1997) 4104-4108
21. Xiaohui, H., Eberhart, R.C., Yuhui, S.: Swarm intelligence for permutation optimization: a case study of n-queens problem. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. IEEE Press, Indianapolis, USA (2003) 243-246
22. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: simpler, maybe better. Evolutionary Computation, IEEE Transactions on 8 (2004) 204-210
23. Robinson, J., Rahmat-Samii, Y.: Particle swarm optimization in electromagnetics. Antennas and Propagation, IEEE Transactions on 52 (2004) 397-407
24. Barchino, R., Gutiérrez, J.M., Otón, S.: 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), Vol. 1. IADIS Press, Virtual (2005) 140-141