A New Sequencing Method in Web-Based Education

Luis de-Marcos, José J. Martínez, José A. Gutiérrez, Roberto Barchino, José M. Gutiérrez

Abstract— **The process of creating e-learning contents using reusable learning objects (LOs) can be broken down in two sub-processes: LOs finding and LO sequencing. Sequencing is usually performed by instructors, who create courses targeting generic profiles rather than personalized materials. This paper proposes an evolutionary approach to automate this latter problem while, simultaneously, encourages reusability and interoperability by promoting standards employment. A model that enables automated curriculum sequencing is proposed. By means of interoperable competency records and LO metadata, the sequencing problem is turned into a constraint satisfaction problem. Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) agents are designed, built and tested in real and simulated scenarios. Results show both approaches succeed in all test cases, and that they handle reasonably computational complexity inherent to this problem, but PSO approach outperforms GA.**

I. Introduction

Brusilovsky [1] 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 [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. 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" [6]. But offering personalized paths to each learner will impose prohibitive costs to these providers, because the sequencing process is usually performed by instructors. So, "it is critical to automate the instructor's role in online training, in order to reduce the cost of high quality learning" [7] and, among these roles, sequencing seems to be a priority.

In this paper an innovative sequencing technique that automates 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. Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) are AI techniques that have shown a good performance for solving a wide variety of problems. So, GAs and PSO are 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 both evolutionary approaches (PSO and GA) for solving the problem. Section 4 presents the results obtained when agents are tested in simulated scenarios as well as 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.

II. 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) [8].

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 [9] (Shareable Content Object Reference Model). Courseware that conforms SCORM's Content Aggregation Model [10] 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

Manuscript received November 14, 2008. This work was supported in part 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, FIT-350101-2007-9, FIT-020100-2008-23, TSI-020302-2008-11) and Plan Avanza program (grant PAV-070000-2007-103), and (3) 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.

Luis de Marcos is with the Computer Science Department, University of Alcalá, Madrid, Spain (corresponding author to provide phone: 34918856651; e-mail: luis.demarcos@uah.es).

José J. Martínez, José A. Gutiérrez, Roberto Barchino, José M. Gutiérrez are with the Computer Science Department, University of Alcalá, Madrid, Spain (e-mails: josej.martinez@uah.es; jantonio.gutierrez@uah.es; Roberto.barchino@uah.es; josem.gutierrez@uah.es).

978-1-4244-2959-2/09/$25.00 © 2009 IEEE

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.

A. 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" [11]. 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:

- IMS "Reusable Definition of Competency or Educational Objective" (RDCEO) specification [12],
- IEEE Learning Technology Standards Committee (LTSC) "Standard for Learning Technology - Data Model for Reusable Competency Definitions" specification (currently an approved standard) [13],
- and HR-XML Consortium "Competencies (Measurable Characteristics) Recommendation" [14]

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

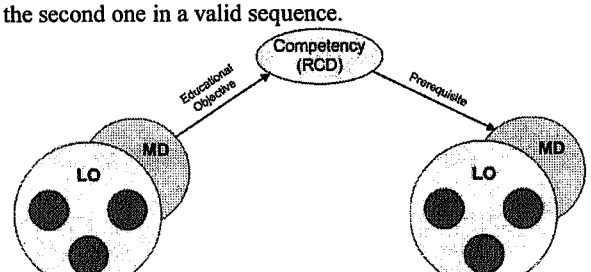

Figure 1. LO sequencing through competencies

Meta-Data (MD) definitions are attached to LOs, and within those definitions references to competencies (prerequisites and learning outcomes) are included. LOM [15] records have been used for specifying LO Meta-Data. LOM element 9, 'Classification', is used to include competency references as recommended in [16, 17]. 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 (see last section) 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.

III. 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. GAs and PSO are AI stochastic population-based computing 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 the agents implementation can be formally specified. Then this implementation is presented.

A. Mathematical Characterization

According to Tsang [18] a CSP is triple *(X,D,C)* where $X = \{x_0, x_1, \ldots, x_{n-1}\}$ is finite set of variables, D is a function that maps each variable to its corresponding domain *D(X)*, and $C_{ij} \subset D_i \times D_j$ 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 (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.

B. GAs for Learning Object Sequencing

Genetic algorithms are an evolutionary computation technique that mimics gene's evolution to solve problems. A random initialized population of individuals is created. Each individual contains a coded state or solution (gene) to the problem; and an iterative process of recombination, mutation and selection is used to evolve population and, simultaneously, the solution. GAs that use specific representation and operators for handling permutations are called permutation GAs or permut-GAs and can be employed to solve constraint satisfaction problems [19].

Permut GA with order recombination, swap mutation and generational replacement with elitism was implemented in order to test its performance for solving the LO sequencing problem. Table 1 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.

Table 1. GA sequencing agent pseudo-code

```
SEQUENCING_AG(input_sequence, μ, m, k, p, n)
BEGIN
  SET population[0] = input_sequence
  Randomly INITIALIZE the rest of the population μ
  EVALUATE each individual
  SET bests = n individuals with best fitness
  SET n_generations = 0
  REPEAT UNTIL (termination criterion satisfied
                or n_generations=m)
    SELECT μ/2 couples using k size tournaments
    Perform an ORDERED RECOMBINATION of the μ/2
couples
    Offspring SWAP MUTATION with probability p
    ELIMINATE DUPLICATES
    survivors SELECTION for the next generation
    PERFORM a generational replacement
      FOR-EACH i in bests
        IF fitness(i) > fitness(best offspring)
          REPLACE a random offspring with i
      END FOR-EACH
  END REPEAT
END
```

Fitness Function. It is critical to choose a function that accurately represents the goodness of a solution [20]. For 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 [21]:

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

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 GA objective is to minimize that function. When an individual 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 $C = \{x_i - x_j > 0 : x_i \in X, x_j \in \{x1,..,xi\}$. 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 GA 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)

Recombination. Parent selection and recombination is implemented in a single step. $\mu/2$ couples (or pairs) are selected using μ tournaments. μ is the population's size and it is an input parameter. k-size tournament with replacement is used (k is an input parameter). The two members of each couple must be different to ensure genetic variation in the offspring. Finally each pair is recombined using order crossover producing μ new individuals.

Mutation. Mutation rate for each individual is implemented as an input parameter (p). Mutation probability is computed just after recombination for each newly individual. Swap mutation is finally performed if probability computing requires it. Note that individual-level mutation (and not gene-level) mutation is used.

Survivor selection. A generational model with elitism is used. The current generation is replaced by its offspring. Elitism is implemented to keep track of the best individuals found so far and to use its genetic material for the next generation. Best n individuals found so far are always kept

in the population. Offspring individuals' that are replaced in order to make room to accommodate 'bests' are randomly selected. The elitism's size n is an input parameter that can range from 0 to μ-1. Elitism implemented in this way requires full population's information. This will make difficult (multiple-agent) distributed processing, in case this will be considered for the future.

Initialization and Termination criteria. The algorithm receives an initial sequence I as an input. This input is used to initialize the first individual. All other individuals are initialized randomly by permuting I. Agent processing stops when a fitness evaluation of an individual returns 0 or when a fixed maximum number of iterations is reached. So the number of iterations was also defined as an input parameter (m). 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.

Duplicate elimination. To avoid genetic drift (quick convergence to the same or very similar individual for all the population), observed in the initial stages of development, a duplicate elimination policy was introduced. Just after recombination and mutation processes each individual is compared with the previous elements in the population. If the genotype is the equal to any of these predecessors a swap mutation is enforced until it differs. Duplicate elimination also requires full knowledge about the population.

C. PSO for Learning Object Sequencing

Particle Swarm Optimization 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 their velocity and cooperate towards finding a solution. There are three levels of best solutions: (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. Original PSO [22, 23] is intended to work on continuous spaces. A version that deals with permutation problems was introduced in [24]. This discrete approach was employed and a full-informed version of the PSO was implemented (table 2) in order to test its performance for solving the LO sequencing problem.

Table 2. PSO sequencing agent pseudo-code

```
SEQUENCING_PSO(input_sequence, w, c1, c2) {
initialize the swarm
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×rnd()×(pbest-currentValue)) +
           (c2×rnd()×(gbest-currentValue))
  NORMALIZE Velocity as
   V_norm = V_new/max(V_new)
  CHECK V_norm limit
   FOR EACH v in V_norm {
    IF(v > length(X))
     v = length(X)
   }
  UPDATE particle value
   FOR i = 1 to length(V_norm){
     IF(rand() < V_norm[i])
      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. Several issues concerning design and implementation were also decided. In the rest of this section each of these issues is discussed and the selection criteria are explained.

The first issue will be the fitness function. A standard penalty function will be used. All aspects referred in section III.B are applicable here.

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 [25] 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:

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

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 number of iterations setting can avoid infinite computing.

Finally, although several tuning mechanisms have been envisaged and tested but only a velocity check policy improves performance [26]. So it was decided to implement this check.

IV. Experimental Results

Both algorithms for LOs sequencing described above were designed and implemented using the object oriented paradigm. We wanted to test their 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 [27], 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 $8{,}9 \times 10^{12}$. This number reflects the number of states (non-feasible solutions) for each feasible solution.

When the test case was established we face the problem of parameter setting for each algorithm. PSO parameters and tuning settings were described in the previous section. As for the GA there were 5 parameters to be tuned. The best configuration depends on each problem [19] so parameters were tuned during this phase. We decided to try different values (ranging from 4 to 6 values) for each parameter. A thorough test of all possible cases would have required 400 executions. In order to reduce this, a 'pivot rule' method was devised. A central configuration (pivot set) was established, and just one parameter (all its values) was tested per round. Further refinement could be achieved if all observed improvements over the initial pivot set are used to determine a new pivot set and the process is repeated. Results from this process return that the optimal set of values for each GA parameter was: population size μ=20, tournament size k= $\mu/3$, mutation rate p=0.5 and elitism size n=$\mu/2$. k and n were deliberately set to be population size (μ) dependent.

When all parameters were set 100 tests were run computing mean fitness values evolution using the best configuration found for each agent (figure 2). Both agents converge, but PSO approach outperforms GA. Mean fitness values to reach a solution (table 1) also support this argument.

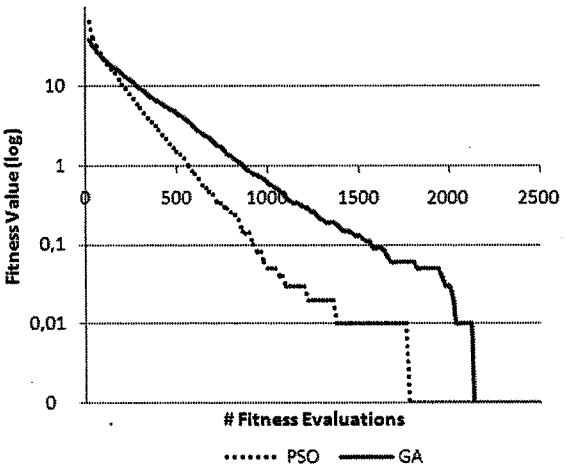

Figure 2. Agents performance in a real world problem.

Table 3. Mean number of fitness

Technique	Mean fitness evaluations
GA	1002
PSO	641

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 for each agent and mean values were computed. Figure 3 shows the results and it also supports the argument that PSO outperforms GA. It could also be inferred that both agents handle 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.

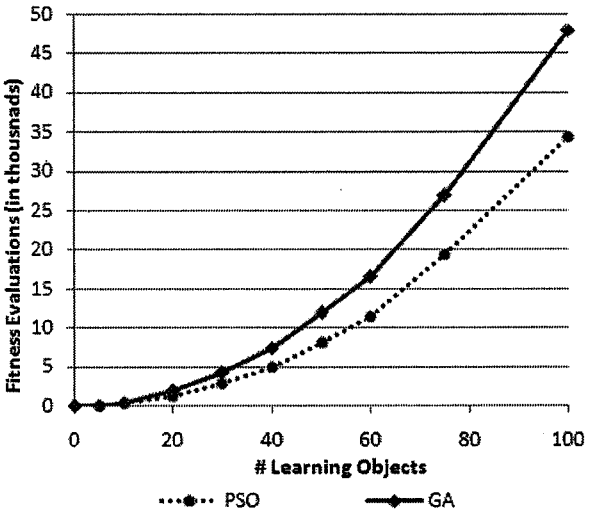

Figure 3. Agents scalability in simulated scenarios

V. 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 two agents that perform automatic LO sequencing through competencies in order to study its completeness and performance. 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. A GA that handles permutation problems has been developed and the PSO for permutation problem has been extended to LO sequencing problem. Results show that both agents succeed in solving the problem and that PSO implementation outperforms GA agent.

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 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 is depicted in figure 4. Sequencing process can be complemented with gap analysis process and competency learner modeling techniques to build personalized courses. These courses could also be SCORM [10] compliant, so they could be imported to current LMSs.

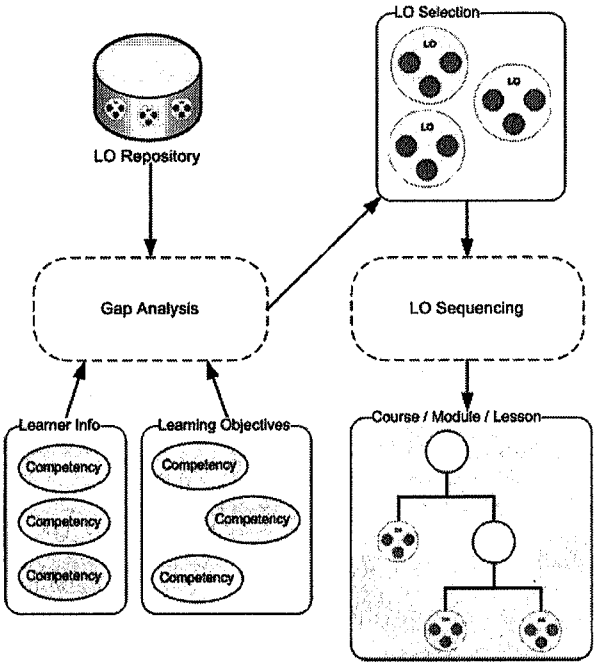

Figure 4. Competency-driven content generation model

Finally, other AI techniques should be analyzed to test its performance for solving the LO sequencing problem. Particularly, Ant Colony Optimization (ACO) [28]. Exact techniques may also be considered. We plan to design and build intelligent sequencing agents using these techniques and check its results against PSO and GA implementation performance.

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

[7] A. Barr, "Revisiting the -ilities: Adjusting the Distributed Learning Marketplace, Again ?," *Learning Technology Newsletter*, vol. 8, pp. 3-4, January/Aprl 2006.

[8] D. A. Wiley, "Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy," in *The Instructional Use of Learning Objects*, D. A. Wiley, Ed., 2000.

[9] ADL, "Shareable Content Object Reference Model (SCORM). The SCORM 2004 Overview," Advanced Distributed Learning (ADL) Initiative, 2004.
[10] ADL, "Shareable Content Object Reference Model (SCORM). The SCORM 2004 Content Aggregation Model," Advanced Distributed Learning (ADL) Initiative, 2004.
[11] J. Wilkinson, "A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM," in *IEEE International Conference on Advanced Learning Technologies, 2001. Proceedings*, 2001, pp. 205-208.
[12] IMS, "Reusable Definition of Competency or Educational Objective - Information Model," IMS Global Learning Consortium, 2002.
[13] IEEE, "Learning Technology Standards Committee (LTSC). Draft Standard for LearningTechnology - Data Model for Reusable Competency Definitions," IEEE, 2007.
[14] HR-XML, "Competencies (Measurable Characteristics) Recommendation," HR-XML Consortium, 2006.
[15] IEEE, "Learning Technology Standards Committee (LTSC). Learning Object Metadata (LOM). 1484.12.1," IEEE, 2002.
[16] 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.
[17] IMS, "Reusable Definition of Competency or Educational Objective - XML Binding," IMS Global Learning Consortium, 2002.
[18] E. Tsang, *Foundations of Constraint Satisfaction*. London: Academic Press, 1993.
[19] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Berlin (Germany): Springer-Verlag, 2003.
[20] J. Robinson and Y. Rahmat-Samii, "Particle swarm optimization in electromagnetics," *Antennas and Propagation, IEEE Transactions on*, vol. 52, pp. 397-407, 2004.
[21] L. Schoofs and B. Naudts, "Ant colonies are good at solving constraint satisfaction problems," in *Proceedings of the 2000 Congress on Evolutionary Computation.*, La Jolla, CA, 2000, pp. 1190-1195.
[22] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science. MHS '95.*, Nagoya, Japan, 1995,

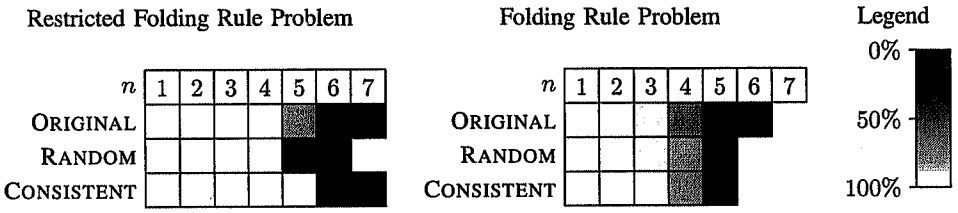

Fig. 5. Percentage of successful runs in our experiments

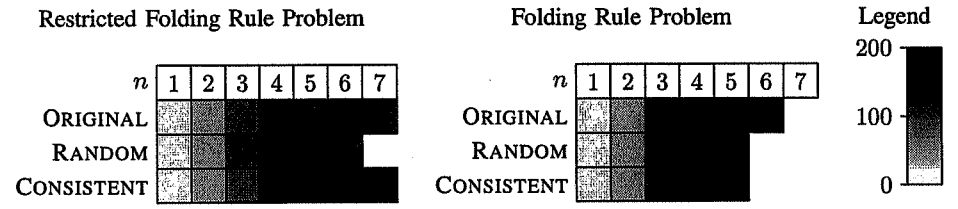

Fig. 6. The Average number of Evaluations pr Success (AES) in our experiments.

Strategy	Success	AES
ORIGINAL	12%	4,744
RANDOMSIGN	12%	4,810
CONSISTENT	12%	5,176

TABLE I
SUMMARY OF ALANINE DIPEPTIDE EXPERIMENTS.

The alanine dipeptide experiments are summarized in Table I and indicate that the altered recombination strategies have less impact on problems from computational chemistry. As can be seen from the table we actually perform more evaluations before reaching the desired goal when using the two altered strategies and this experiment is therefore contradictory to the previous, indicating that the original DE strategy is more robust across problems with more complex fitness landscapes.

VII. CONCLUSION

To sum up, our experiments indicate that the choice of recombination strategy has a large effect on the percentage of successful runs in simple cases, but that it has little effect on the number of fitness evaluations in these. Our experiments furthermore illustrate that the choice of optimization problem can have a large impact on the observed difference between the recombination strategies and in some cases it can lead to contradictory conclusions. We cannot confirm nor refute our hypothesis that angles should not be handled naïvly.

Our studies poses two interesting questions: (1) is testing new ideas on sandbox problems a valid evaluation method and (2) is over engineering functions to specific optimization scenarios a problem in the field. It is our hope that others in the field will investigate these two interesting questions.

REFERENCES

[1] K. Price and R. Storn, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," *International Computer Science Institute-Publications*, Jan 1995.

[2] ——, "Differential evolution a simple evolution strategy for fast optimization," *Dr.Dobbs Journal*, no. 22, pp. 18–24, 1997.

[3] U. K. Chakraborty, *Advances in Differential Evolution*. Springer Publishing Company, Incorporated, 2008.

[4] R. Storn, "Differential Evoultion Homepage," http://www.icsi.berkeley.edu/~storn/code.html.

[5] K. Zielinski, X. Wang, and R. Laur, "Comparison of adaptive approaches for differential evolution," *Parallel Problem Solving from Nature – PPSN X*, pp. 641–650, 2008.

[6] J. Handl, S. Lovell, and J. Knowles, "Investigations into the effect of multiobjectivization in protein structure prediction," *Parallel Problem Solving from Nature – PPSN X*, pp. 702–711, 2008.

[7] N. O'Boyle, C. Morley, and G. Hutchison, "Pybel: a python wrapper for the openbabel cheminformatics toolkit," *Chemistry Central Journal*, Jan 2008.

Total páginas