International Journal of Engineering Education Vol. 31, No. 4, pp. 1092–1105, 2015
Printed in Great Britain

0949-149X/91 $3.00+0.00

Parliamentary Optimization to Build Personalized Learning Paths: Case Study in Web Engineering Curriculum*

LUIS DE-MARCOS, ANTONIO GARCIA-CABOT, EVA GARCIA-LOPEZ and JOSE-AMELIO MEDINA
Computer Science Department. University of Alcalá. Edificio Politécnico. Campus Universitario. Ctra Barcelona km 33.6. Alcalá de Henares. Madrid. Spain. E-mail: luis.demarcos@uah.es, a.garciac@uah.es, eva.garcial@uah.es, josea.medina@uah.es

In this paper we present a practical application of POA (Parliamentary Optimization Algorithm) for creating personalized learning paths in online learning. The objective of building a personalized learning path is to produce a suitable sequence of learning units for a student to work with. We present and tune the parliamentary metaheuristic for a practical instance of the sequencing problem in a web engineering master programme and compare it with standard versions of other well established metaheuristics (PSO and genetic algorithms). Results suggest that permut-POA deals satisfactorily with sequencing problems and it is easy to fine tune, and also that it outperforms the other optimizers.

Keywords: courseware sequencing; personalized learning; e-learning; parliamentary optimization algorithm; particle swarm optimization; genetic algorithm; web engineering education

1. Introduction

Parliamentary Optimization Algorithm (POA) [1, 2] is a stochastic metaheuristic inspired in the behavior observed in political parties when trying to gain control over parliaments in head elections. The process is implemented as follows: firstly several individuals are randomly initialized, and then arbitrarily divided into groups. The most fitted individuals of each group are designated as candidates. Then the groups participate in an iterative process that involves intra-group and inter-group fitness-based competitions. During intra-group competition, the candidates of each group bias regular members who then move to new positions closer to candidates thus exploring the most promising areas of the search space. After biasing, new candidates are selected again among the most fitted individuals. During inter-group competition, groups stochastically form alliances to merge and less powerful (fitted) groups can be removed (disappear). The process finishes when a predefined ending criterion is met, and then the best candidate of all groups (i.e. the best solution) is returned. POA is then a population-based metaheuristic that evolves a solution over an iterative process.

POA has demonstrated to be competitive for numerical optimization, even outperforming other well-known and well-studied stochastic approaches such as Genetic Algorithms (GA) [1] and Particle Swarm Optimization (PSO) [2]. Performance improvement is measured through a set of standardized test-bench functions in terms of the quality of the final solution and in terms of efficiency (time) required to get that solution (e.g. calls to the fitness function). Numerical optimization is an interesting and challenging field with many potential applications, but combinatorial optimization is also another demanding area with important applications that needs to be studied [3].

An interesting application of combinatorial optimization is courseware sequencing, which is a discipline that tries to personalize and adapt learning contents to the needs of individual students. It aims to create a correct sequence of learning units (courses, subjects, lessons, etc.) according to the prior knowledge of the student, i.e., adapted to her competencies. We use concept of competence [4] to define both requisites and learning outcomes for each individual learning unit. Such an approach enables domain-independent sequencing where the problem of finding a feasible sequence for the learner can be represented as a classical permutation Constraint Satisfaction Problem (CSP) [5].

This paper introduces a new version of POA (permut-POA) designed to deal with permutations and constraint satisfaction problems for solving the problem of courseware sequencing. Section 2 introduces courseware sequencing within the larger framework of courseware engineering. Section 3 describes the parliamentary metaphor underlying POA algorithm and our adaptation for permutation constraint satisfaction. Sections 4 (experimentation) and 5 (results) present the experiments carried out to fine-tune the optimizer and to test its performance in a particular test case. Section 6 presents a comparative analysis of POA's performance with two other evolutionary approaches (particle swarms and genetic algorithms) for this particular instance of the sequencing problem. And finally, Section 7 presents conclusions and future work.

* Accepted 7 March 2015.

Table 1. Example of a web development course represented with prerequisites and learning outcomes

Learning unit	Prerequisites	Learning outcomes
1.Introduction to Web Developing	–	Learn the basics of web development
2. HTML	Basic web development	HTML knowledge
3. CSS	HTML knowledge	Build and use CSS style sheets
4. JavaScript	HTML knowledge	Develop JavaScript programs
5. HTML 5	HTML knowledge, develop JavaScript programs	Building HTML5 web pages
6. Java EE	HTML knowledge	Develop Java EE programs

2. Courseware sequencing

Courseware engineering is a broad discipline encompassing a set of practices, methods and tools, whose primary goal is to improve the effectiveness and efficiency of courseware production [6]. More specifically, it can help in the life-cycle stages of designing, building and maintaining course materials. Some of the tasks performed in these stages can be performed in a semi-automatic way, while others can be carried out automatically. Courseware sequencing (also referred as curriculum sequencing) is a technique widely used in courseware engineering that aims to provide the student with the most suitable individually planned sequence of knowledge units to learn and the sequence of learning tasks (examples, questions, problems, etc.) to work with [7]. Courseware sequencing can be used at different stages to design, build, maintain and test learning paths. It can be used to personalize and adapt contents to the needs of each particular learner [8].

When it comes to curriculum sequencing standard typologies further distinguish between domain-dependent and domain-independent approaches. On the one side, domain-dependent approaches rely on a domain model to represent the knowledge of a specific domain. This model is used to tailor contents to learners' needs. Modeling approaches and sequencing strategies have been vastly studied by the fields of intelligent tutoring systems and adaptive hypermedia [9]. Domain-dependent courseware sequencing can fine-grain personalization but the domain model has to be built and updated for every area in which sequencing is needed and this is their main disadvantage. On the other side, domain independent approaches aim to work in every possible domain. To do these they mostly rely on standards that can unambiguously define dependencies between learning units or that can be used to infer such dependencies. Domain-independent sequencers have come mostly due to the efforts of e-learning standardization and interoperability communities [5, 10]. Despite their wide applicability, the effectiveness of independent-domain sequencing depends on the quality of the description attached to learning resources (metadata) and their uniformity, especially in terms of size (granularity).

Competencies can be used for domain-independent sequencing. There are many definitions of term competency coming from different areas including pedagogy, human resources management and computer science. In our work competencies are defined as 'multidimensional, comprising knowledge, skills and psychological factors that are brought together in complex behavioral responses to environmental cues' [4]. This definition emphasizes that competencies are a multidimensional set of factors (not only knowledge) and that they are employed (brought together) in real or simulated contexts (environments). A reusable (standardized & interoperable) definition of competency can be used as a prerequisite or a learning outcome [11] of a given learning unit (course, subject, lesson, etc.). A prerequisite is defined as a competency that the learner needs to have in order to undertake the learning unit. A learning unit can have one or more prerequisites (or none at all). A learning outcome could be defined as the set of competencies obtained by the students when they complete a learning unit. Table 1 presents an example of a set of learning units and their corresponding prerequisites and learning outcomes.

Courseware sequencing will then try to build a personalized sequence of learning units taking into account the competencies defined. For the example shown in Table 1 different sequences that meet all requirements can be built, as shown in Table 2.

The sequencing problem may seem easy but with a significant number of learning units it actually becomes hard to solve. The question of finding a correct sequence can be represented as a classical Constraint Satisfaction Problem (CSP). Prerequisites and learning outcomes define the set of con-

Table 2. Some valid sequences for the web developing program

Sequence #1	Sequence #2	Sequence #3
1. Introduction. . .	1. Introduction. . .	1. Introduction. . .
2. HTML	2. HTML	2. HTML
3. CSS	4. JavaScript	6. Java EE
4. JavaScript	5. HTML 5	3. CSS
5. HTML 5	3. CSS	4. JavaScript
6. Java EE	6. Java EE	5. HTML 5

straints to be satisfied. The space of solutions comprises all possible sequences ($n!$ will be its size, total number of states for n learning units), and a (feasible) solution is a sequence that satisfies all established constraints. Permutations of learning units within the sequence are the operations that define transitions among states. A new sequence is created if one or more swaps between pairs of learning units are performed. Each swap generates a new sequence that has to be checked to make sure it does not violate any constraint. Therefore we face a permutation constraint satisfaction problem which is a concrete type of CSP in which operations are defined in terms of permutations between pairs of objects. Different approaches have been proposed to solve the sequencing problem modeled as a permut-CSP: Genetic Algorithms [11], Particle Swarm Optimization [12] and other classical approaches like heuristic and local search [13]. In this paper a new approach based on an adaptation of POA for solving the courseware sequencing problem is proposed. The following section presents the modification undertaken in POA for tackling this problem.

3. The parliamentary metaphor for permutation constraint satisfaction

Parliamentary metaphor is inspired in the behavior observed in political parties when trying to gain control over parliaments in head elections [1]. It is a stochastic metaheuristic algorithm that was initially designed for numerical optimization and thus works in a continuous space [2]. POAs original implementation is presented in Fig. 1.

In order to apply POA to solve permut-CSPs some of their internal workings have to be redefined [14] so that it can operate in a discrete space in which individuals are also permutation sets. In the following sections the POA adaptation to solve permut-CSP is described.

3.1 Redefining biasing

Initialization and inter-group competition do not require special modifications to work in this new landscape. Initialization deals with partitioning the population in groups and selecting the most fitted individuals. Inter-group competition deals with merging and deleting groups stochastically. However, as for intra-group competition the process of biasing regular members towards candidates needs to be redefined to work in a permut-CSP. Original POA uses a linear fitness-proportional weighting towards all candidates in the group to bias regular members. Other metaheuristics like PSO also use similar methods that have been widely discussed but this approach is not valid for permutations. We suggest instead using fitness proportional selection to stochastically permute each position in the tuple towards one candidate. Fitness proportional selection was originally introduced for genetic algorithms [15] and can be employed in the different selection processes of any evolutionary algorithm, i.e., parent selection and survivor selection. Premature convergence in unbalanced population distributions [16] has been indicated as one of the main drawbacks for this method; so alternative methods like ranking selection and stochastic universal sampling (SUS) [17] have been introduced. But quick convergence may not be a problem in this new definition of POA because selection will be stochastically performed for each dimension of the solution.

Let T be a candidate solution tuple, the probability for each position, $t_j \in T$, in the regular member T to move towards a candidate $c \in \varphi$ can be calculated according to equation 1. All candidates in addition to the current individual form the pool of elements that are considered for selection. Probabilities are computed once and then random numbers are generated for each dimension to determine the bias. If the movement of any position is finally required, the position will be set to the value

```
POA (N, L, θ, λ, pm, γ, pd)
  Initialize population
    Partition of the population in N groups of L members
    Pick θ most fitted individuals as candidates for each group
  repeat
    Intra-group competition
      Bias regular members towards candidates θ
      Reassign candidates
      Compute power for each group
    Inter-group competition
      Merge λ strongest groups with probability pm
      Remove γ weakest groups with probability pd
  until(stopping condition met)
  return the best candidate

end
```

Fig. 1. Pseudo-code of the Parliamentary Optimization Algorithm (from [2]).

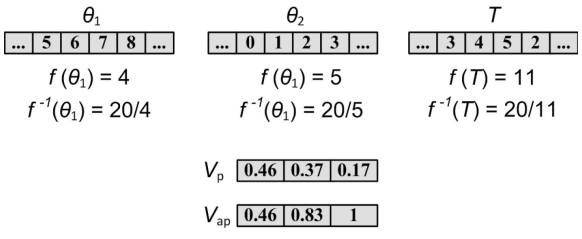
of the same position in the candidate selected by swapping values. A similar approach has been described and successfully employed for PSO [18].

$$t_j^{swap \to c} = \frac{f(c)}{\sum_{i=0}^{\theta} f(\theta_i) + f(T)} \quad (1)$$

f is the fitness function used to assess the goodness of solutions. A common choice when dealing with CSP is a standard penalty function that computes the number of constraints violated by a candidate solution. A permut-CSP with a standard penalty function is a minimization problem, so it is necessary to use an inverse weighting function in order to assign larger probabilities to individuals with lower (better) fitness. The inverse fitness function is presented in equation 2. Then equation 1 will return a fitness proportional probability if every call to f is substituted for a call to f^{-1}.

$$f^{-1}(T) = \frac{\sum_{i=0}^{\theta} f(\theta_i) + f(T)}{f(T)} \quad (2)$$

As for the implementation of members' biasing, probabilities for each candidate were stored in an intermediate vector (V_p), and another vector with aggregated probabilities (V_{ap}) was also created to determine value ranges for evaluating random numbers. Fig. 2 shows an example which represents a part of the biasing process in a hypothetical job scheduling problem. Two candidates (θ_1 and θ_2) and the current individual T are considered. They are denoted as vectors, whose positions contain numbers representing the name of the task. Sample fitness values are considered to calculate inverse fitness values and both probability vectors. Biasing is exemplified in the final table of the figure. Three random numbers are generated to decide the swap for each position. The candidate selected towards which bias is actually performed, and the resulting state of the individual T are also displayed.



Position	*Random number*	*Biasing towards*	*T*
T_i	0.1	θ_1 ... 5 6 7 8 ...	... 5 4 3 2 ...
T_{i+1}	0.9	none	... 5 4 3 2 ...
T_{i+2}	0.7	θ_2 ... 0 1 2 3 ...	... 5 4 2 3 ...

Fig. 2. An example of member biasing for permutation problems.

Please note that in this discrete version regular members always update their position. Our aim was to improve individuals' mobility in the new landscape. It should be noted that many discrete spaces may contain plateaus, i.e., large areas in which all solutions contain values with identical or similar fitness values. If regular members are compelled to move, wider search areas will be explored at no especial higher expense (new positions are always evaluated).

3.2 Considering mutation

Evolutionary algorithms introduced mutation since their beginning [19, 20] as a way to model similar processes occurring in nature. The basic idea is to randomly alter an individual gene feature to make the individual uniquely different from each of its antecessors, thus increasing the diversity of the population. Mutation is very important to the extent that some evolutionary algorithms rely heavily or even exclusively on mutation procedures to generate new candidate solutions, e.g., in evolution strategies [21].

To increase population diversity, a simple mutation mechanism was introduced in the permut-POA. Mutation is implemented just after regular members complete biasing towards candidates. Mutation rate for each individual is implemented as an input parameter (p) that is used as an individual-level mutation (not gene-level mutation).

3.3 Duplication elimination policy

In the initial stages of development a genetic drift (i.e. quick convergence to the same or very similar individual for all the population) was observed, so a duplicate elimination policy was introduced to avoid it. When the biasing and mutation processes have been performed, each individual is compared to the previous elements in the group. If there is any other individual equal to the new one, a swap mutation is performed until it differs from all of them. This implementation can imply huge computational costs when POA faces massive populated groups and it will also hinder any future distributed POA approach because full information about the group is required to implement it. Therefore, a new boolean parameter (*de*) was introduced to enable or disable it.

Fig. 3 presents the pseudo-code of the final version of POA modified to address permutation problems. Differences with the original implementation are boldfaced. Permut-POA has been used for solving the problem of courseware sequencing as presented in this paper. The next section explains the test case designed for carrying out the experiment with this new metaheuristic.

```
Permut-POA (N, L, θ, p, λ, pm, γ, pd)
  Initialize population
    Partition of the population in N groups of L members
    Pick θ most fitted individuals as candidates for each group
  repeat
    Intra-group competition
      Bias regular members towards candidates
      Mutate members with probability p
      if de is true Eliminate duplicates
      Reassign candidates
      Compute power for each group
    Inter-group competition
      Merge λ strongest groups with probability pm
      Remove γ weakest groups with probability pd
  until(stopping condition met)
  return the best candidate

end
```

Fig. 3. Pseudo-code of POA for combinatorial optimization (permut-POA).

4. Experimentation and testing

4.1 Test case

The permut-POA was implemented using the object oriented paradigm. We wanted to test its performance in a real scenario, so a problem concerning course sequencing for a Web Engineering Master program in our institution was chosen for the test. The Web Engineering Master program (Fig. 4) comprises 24 courses (subjects) grouped into the following categories:

- Basic courses (7) that must be taken before any other kind of course. There may be restrictions between two basic courses, e.g., "HTML" course must precede "Basic 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, e.g., "Software Engineering" course must precede "Development Methodologies" course.
- Elective courses (7). Additional constraints with respect to any other course may be set.

All courses have 4 or 6 ECTS credits. In this Master degree the students have to complete 120 ECTS in total. A feasible sequence must have 23 courses (learning units) satisfying many constraints. Please note that the students must choose just one learning path (Java or .NET) and take all the courses on that path. Students must choose six out of the seven elective courses presented in Fig. 4. The five courses of the non-selected path are also eligible as elective subjects. There are 56 constraints among different courses. Just to offer an idea, there are four constraints between basic courses, four constraints between compulsory courses, four constraints between itinerary courses and just one constraint in the elective courses. Most of the constraints appear in the relations that take place between courses belonging to different groups: 14 constraints from compulsory courses to basic courses, 7 constraints from itinerary courses to basic courses, 21 constraints from elective courses to basic courses, and one constraint from one elective course to one compulsory course. The graph for showing all courses and constraints is very complex. Fig. 4 just represents groups of courses. It should be noted that relations between specific courses are not shown in the figure in order to avoid confusion because it would made the figure difficult to read. Calculating the exact number of feasible solutions is also difficult, so an estimation has been used: we have estimated that the relation among feasible solutions and total solutions is in the order of 8.9×10^{12}. This number reflects the number of states (non-feasible solutions) for each feasible solution. This means that the chance of a random sequencer selecting a feasible sequence is $1/(8.9 \times 10^{12})$.

4.2 Parameter tuning

POA has many parameters and there are no studies, as far as authors are concerned, about them. Original work on POA offers a set of values for parameters without justifying their selection. This is, in our opinion, a serious drawback for POA since it is essential for a good algorithm performance. Practitioners that have to select a technique for a particular combinatorial problem will almost certainly choose one in which a substantial background work about parameter control (including best practices) exists. To try to mitigate this problem, and always focusing on a practitioner's stance, we have conducted a preliminary study to try to set best practices for parameter selection in permut-POA.

Parameter control techniques describe how para-

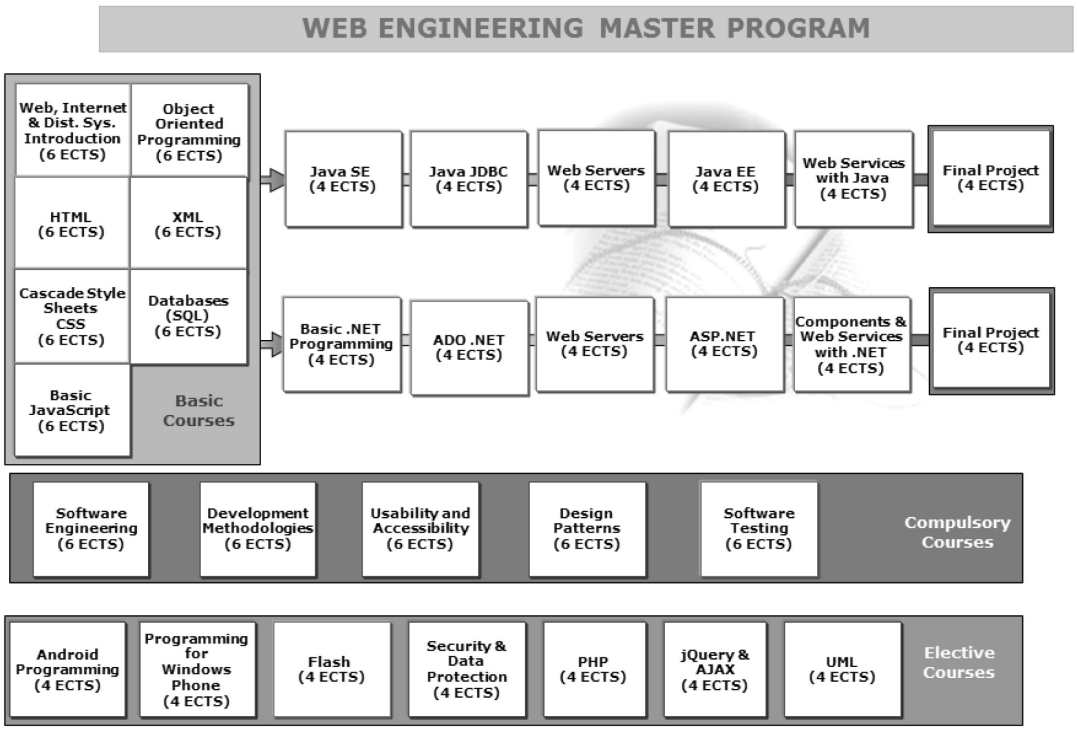


Fig. 4. Web Engineering Master Program.

meters change during the execution of an evolutionary algorithm. They can be classified into three categories [22]: Deterministic parameter control, adaptive parameter control and self-adaptive parameter control. The deterministic approach is the easiest way. It is based on a deterministic rule that sets parameters in a fixed and predefined way. If parameters are set before running the algorithm then the process is also known as parameter tuning. Adaptive control techniques consider feedback from the search; and the self-adaptive approach also considers feedback but in this case parameters are evolved along with the solution. Parameter tuning may not be the best approach in terms of algorithm performance but in many times it is the preferred choice for two main reasons. The first reason is that it is the easiest and the fastest solution to the problem of setting parameters values (in terms of development cost). In many occasions it is impossible to test all possible cases for all parameters, but using approximation techniques is easy to find a set of parameter values that works efficiently. And the second reason is closely related to that fact: evolutionary algorithms are usually quite flexible and it is easy to find a configuration of parameters that works; and with a little bit of extra work it is even possible to find a good configuration that performs well for a wide range of problems. It is then not difficult, just costly in terms of time, to find the best configuration but it is not so costly to find an efficient configuration. So parameter tuning is always a reasonable choice, in terms of effort, to find a good configuration of parameters [23]. Because of this, parameter tuning was chosen to set the parameters for the permut-POA metaheuristic.

Table 3. Values selected for each parameter (pivot values in bold)

Parameter	Symbol	Values
Number of groups	N	2, **3**, 5, 10
Group size	L	5, **10**, 20
Candidates per group	θ	1, **2**, 3, 5
Member weighting constant	m	0.5, **1**, 1.5, 2
Candidate weighting constant	n	0.01, **0.1**, 0.5, 1
Merge probably	p_m	0, **0.01**, 0.1
Deletion probably	p_d	0, **0.01**, 0.1
Groups to be merged	λ	**2**, 3
Groups to be deleted	γ	**1**, 2
Mutation probability	p	0, 0.01, **0.1**, 0.5, 1
Duplicate elimination policy	de	**0**, 1

The process to tune the parameters is a hill climbing in the parameter space. For each parameter a set of representative values were selected for testing (Table 3). We tried to cover the widest range of (reasonable and meaningful) possibilities. Then, every test value of each parameter was tested keeping all other parameters constant in a kind of pivoting rule. The central value of each parameter was initially selected for the pivot set, and when the number of values was even then the selection was arbitrary between the two central values. Selected pivoting values are boldfaced in Table 3. Then a new pivot set is selected with the best value for each parameter and the whole process is repeated.

Table 4. Descriptive statistics of the number of calls to the fitness function for each experiment (100 runs per experiment)

Value	Mean	Std Error	Std Dev	Minimum	Median	Maximum
Parameter: Number of groups (*N*)						
2	874.1	36.7	366.6	276.0	812.0	2500.0
3 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
5	1459.0	49.6	496.1	530.0	1400.0	3050.0
10	2664.6	85.1	850.7	900.0	2649.0	5300.0
Parameter: Group size (*L*)						
5	909.2	36.9	368.6	253.0	841.0	2097.0
10 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
20	1263.1	47.4	474.0	432.0	1201.0	3042.0
Parameter: Candidates per group (θ)						
1	1012.3	33.8	337.5	489.0	934.5	1963.0
2 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
3	1120.9	53.6	536.1	324.0	1085.0	3362.0
5	1111.4	48.9	488.7	330.0	977.5	2400.0
Parameter: Member weighting (*m*)						
0.5	1185.7	62.0	619.6	318.0	1045.0	3726.0
1 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
1.5	1085.6	40.4	403.8	294.0	1030.0	2350.0
2	1133.8	49.5	495.2	366.0	1014.0	2382.0
Parameter: Candidate weighting (*n*)						
0.01	1092.6	49.4	494.4	368.0	1026.0	2834.0
0.1 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
0.5	1038.4	46.4	463.7	356.0	939.0	2742.0
1	1043.7	39.1	391.0	366.0	954.0	2734.0
Parameter: Merge and deletion probability (p_{md})						
0	1199.5	44.9	448.8	462.0	1182.0	3078.0
0.01 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
0.1	795.1	36.1	361.4	270.0	673.0	2054.0
Parameter: Groups to be merged (λ)						
2 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
3	1151.8	49.1	458.2	386.0	998.0	2390.0
Parameter: Groups to be deleted (γ)						
1 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
2	1092.4	46.6	466.0	348.0	1038.0	2854.0
Parameter: Mutation probability (*p*)						
0	1068.8	43.0	430.4	288.0	1005.0	2648.0
0.01	1117.5	47.7	476.6	414.0	1014.0	3222.0
0.1 (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
0.5	1022.8	37.5	375.4	318.0	962.0	2166.0
1	891.9	36.4	363.6	294.0	812.0	2158.0
Parameter: Duplicate elimination policy (*de*)						
0 (disabled) (pivot)	1031.3	35.5	355.5	318.0	990.0	2460.0
1 (enabled)	1234.2	49.8	498.2	300.0	1120.0	2640.0

Table 5. Results of the General Linear Model Analysis

Parameter	Symbol	F	*p*-value
Number of groups	N	186.54	0.000
Group size	L	20.29	0.000
Candidates per group	θ	0.42	0.742
Member weighting constant	m	0.88	0.454
Candidate weighting constant	n	0.25	0.858
Merge and Deletion probably	p_m, p_d	32.43	0.000
Groups to be merged	λ	3.03	0.083
Groups to be deleted	γ	0.21	0.648
Mutation probability	p	4.51	0.001
Duplicate elimination policy	de	8.99	0.003

5. Results

Thirty-three experiments were carried out with the test case of the Web Engineering Master program to tune the parameters for the permut-POA. The number of calls to the fitness function was computed and used to compare the different configurations. Each experiment was executed 100 times. Descriptive statistics (Table 4) suggest that different values in several parameters can improve the performance of the metaheuristic for this particular test case.

Then a General Linear Model (GLM) was used to determine what parameters had a relevant influence in the algorithm performance. Results (Table 5) of six (out of 11) parameters (N, L, p_m, p_d, p and de) returned a significant p-value suggesting that their values influence performance. All other parameters (θ, m, n, λ and γ) did not return any significance. It should be noted that relevant parameters refer to the population size (N, L) and its evolution (p_m, p_d) along with the modifications introduced, i.e., mutation probability (p) and population elimination policy (de).

For each parameter that was found to be relevant, a one way analysis of variance (ANOVA) was carried out to determine the optimal values among all candidate values. It shall be noted that data for all configurations do not follow a normal distribution but rather seem to fit a lognormal distribution. Therefore a logarithmic transformation was applied and Kolmogorov-Smirnov tests were used to confirm goodness of fit to a normal distribution before running parametric tests. Fig. 5 shows the results obtained in the ANOVA for the parameter N (number of groups). Observed performance was better when there are fewer groups and a value of 2 (two parliamentary groups) returns a significantly better performance. Results obtained for the parameter L (size of the parliamentary group) are presented in Fig. 6. A smaller group size resulted better in terms of performance. Equal values for parameters p_m (probability of merging) and p_d (probability of deletion) were considered in the experiments carried out. Fig. 7 shows the interval plot for these two parameters. A value of 0.10

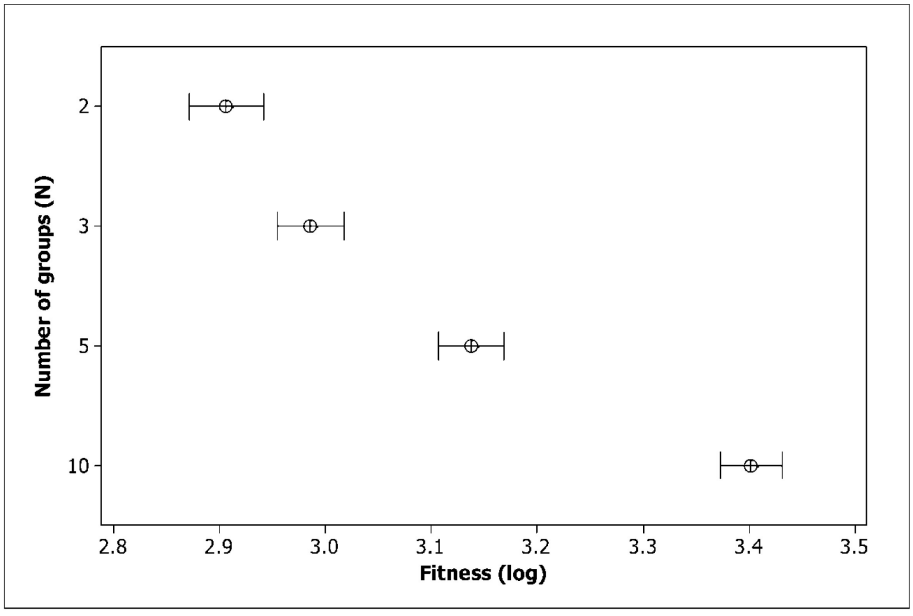


Fig. 5. Confidence intervals of fitness for each value of the number of groups (N) parameter (CI = 95% of the mean).

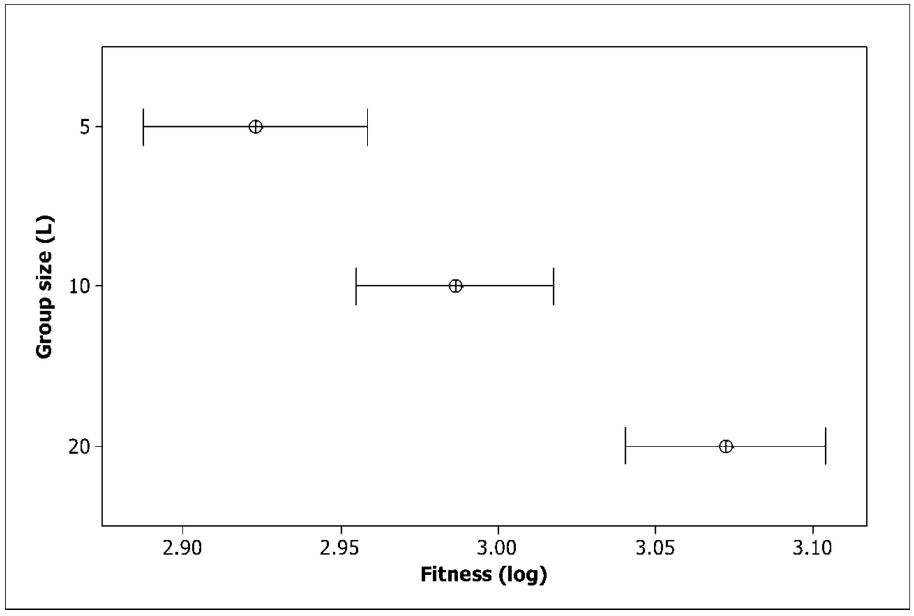


Fig. 6. Confidence intervals of fitness for each value of the group size (L) parameter (CI = 95% of the mean).

(probability of 10% of merging or deleting parliamentary groups) yielded the best performance. Results for the p parameter (mutation probability) suggest that a value of 1 (always mutating) improves algorithm's performance (Fig. 8). The optimal value found for this parameter suggests that the algorithm in some cases converges to local minima (as it has also been reported in other metaheuristics) and finding the solution is difficult without (or with a low) mutation probability. Finally, Fig. 9 shows the results obtained for the *de* parameter, which enables the duplicate elimination policy. A value of 0 (the policy is disabled) improved performance. Duplication elimination policies are usually enabled to avoid quick convergence to local minima because they ensure diversity in the population. But their application can also result in less efficiency in terms of the number of calls to the fitness function in certain landscapes. In this case we consider that for our particular test case population diversity is not so important because that amount of optimal solutions in the solution space is large enough to facilitate convergence even in homogeneous populations and mutation seems to suffice to perform the final steps (swaps) in the candidate solutions.

After completing all ANOVA tests, it was possi-

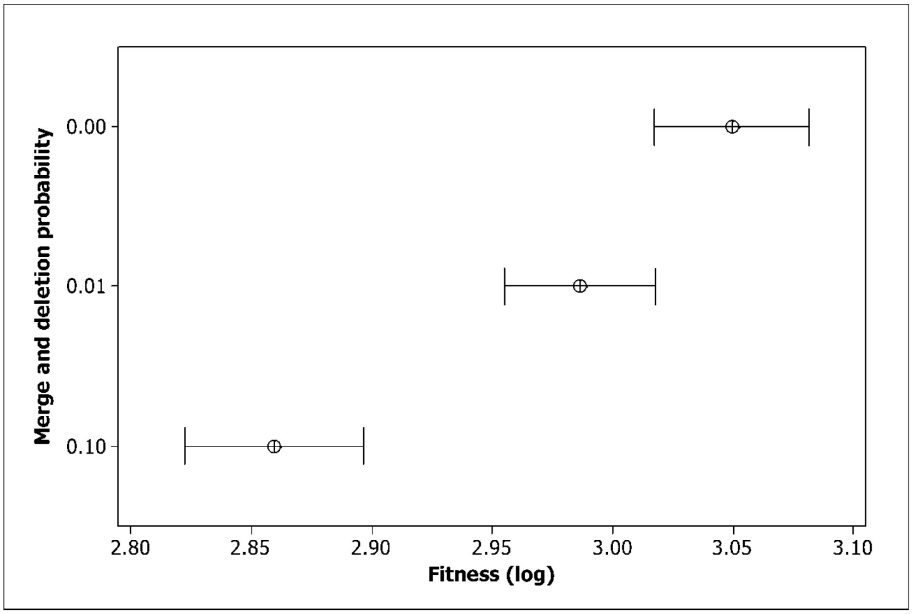


Fig. 7. Confidence intervals of fitness for each value of the merge (p_m) and deletion probability (p_d) parameters (CI = 95% of the mean).

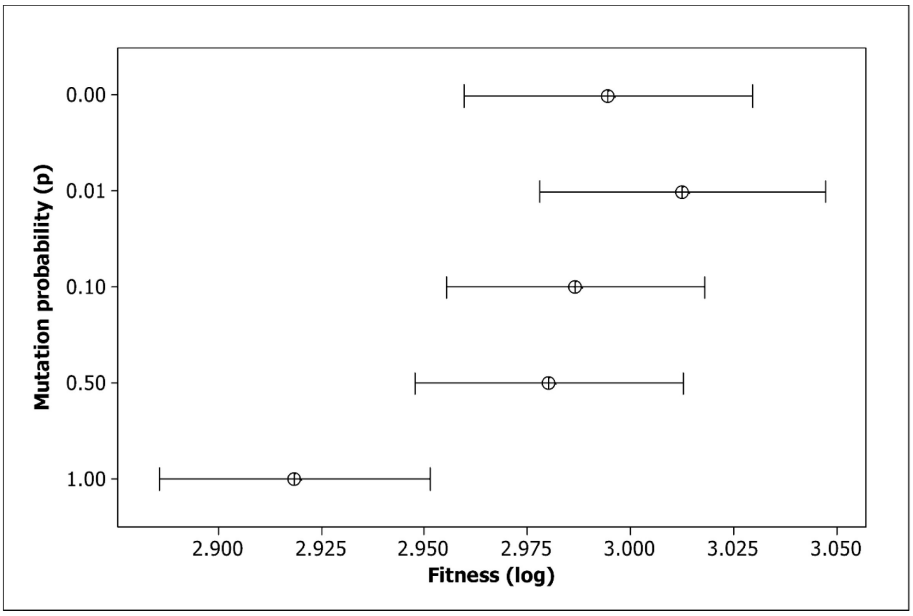


Fig. 8. Confidence intervals of fitness for each value of the mutation probability (p) parameter (CI = 95% of the mean).

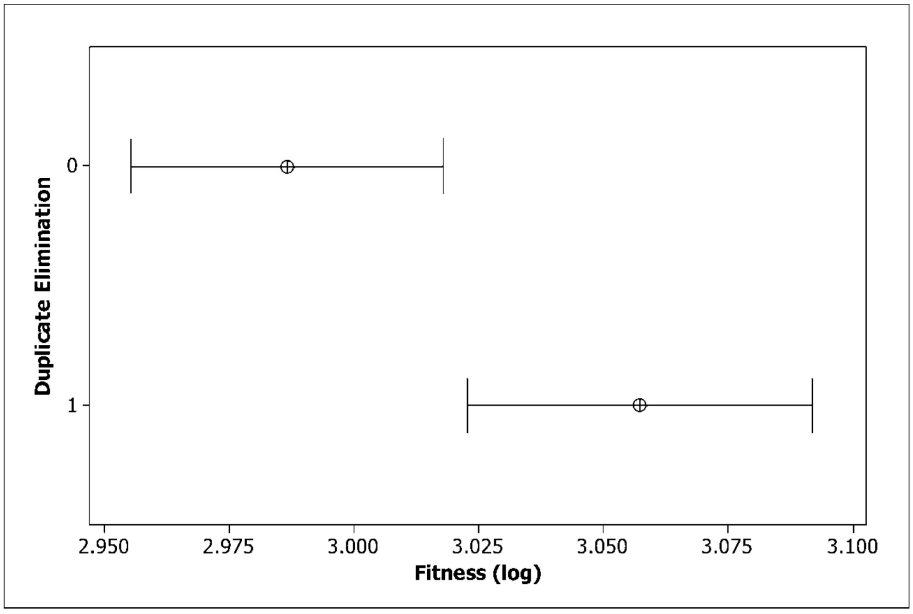


Fig. 9. Confidence intervals of fitness for each value of the duplicate elimination (de) parameter (CI = 95% of the mean).

ble to determine the optimal value for relevant parameters. For each parameter that was found to be not relevant its initial (pivot) value was kept. Therefore, the final optimal values (for the test case) were set as follows: N = 2, L = 5, $\theta = 2$, m = 1, n = 0.1, $p_m = p_d = 0.1$, $\lambda = 2$, $\gamma = 1$, p = 1 and de = 0.

The next section presents a discussion of the results, as well as a comparison between permut-POA and other similar algorithms for solving the same instance of the courseware sequencing problem.

6. Discussion

6.1 Comparing permut-POA with other methods and with existing work

After finding that permut-POA dealt successfully with the test case, the next step was to perform a comparative analysis with other standard evolutionary methods. Genetic algorithms (GA) are a vastly studied subfield in evolutionary computation and they were our first choice. Different approaches to GA can be taken to deal with permutation

problems. Standard typologies usually distinguish between order problems (e.g. job scheduling problem) [24] and adjacency problems (e.g. travel sales person problem) [15]. We developed a permut-GA [11] for order problems that was tested and tuned using the same approach described for the permut-POA.

The second choice was particle swarm optimization (PSO) algorithm, which is a more recent optimizer and has proven its flexibility and efficiency to solve many problems in a wide range of domains [25]. Original PSO [26, 27] is intended to work on continuous spaces. A version that deals with permutation problems (permut-PSO) was introduced in [18]. It outperformed genetic algorithms for the N-Queens problem, so we decided to try this version with all its settings. One important advantage of PSO is that it uses a relatively small number of parameters compared with other metaheuristics. However, much literature about parameters in PSO has been written. Among it, [18] presents a configuration setting that works properly for solving permutation problems, so we decided to follow these recommendations.

In order to perform a comparative analysis each metaheuristic algorithm was executed 100 times. Logarithmic transformation and normality tests were then run. Finally, a one way analysis of variance (ANOVA) was carried out to compare the three. Results (Fig. 10) suggested that Permut-POA outperforms permut-GA and it is competitive with permut-PSO. In order to gain additional insights we performed two additional ANOVA tests comparing POA with the other metaheuristics. p-values returned were less than 0.001 when permut-POA was compared with permut-GA, and 0.003 when permut-POA was compared with permut-PSO. Therefore, results confirmed that permut-POA also outperformed permut-PSO (99.7% CI).

6.2 Relevance for engineering education

Our sequencer was tested in one test case in engineering education, a web engineering master program in which it was used to find a valid path of courses. Permut-POA is designed to operate in domain-independent scenarios so it is potentially useful for any field of education. However we think that it is especially relevant for engineering education because here we can find instances of new and ever evolving disciplines in which there are no predefined paths. It can be particularly useful in computer science and computer engineering education, where besides continuous changes, we can also find multiple open repositories with thousands of learning resources properly annotated with metadata [10].

The sequencer presented in this work is designed to improve the effectiveness and efficiency of courseware production. It could be used at different stages of the life-cycle of courseware for designing, building and maintaining course materials. Potential users of courseware sequencing are students, course and curriculum designers and management. Students can use sequencing to build their own learning paths or get recommendations about them. Course designers can also use it to automate the process of courseware building and offer greater chances of personalization to learners. Sequencers

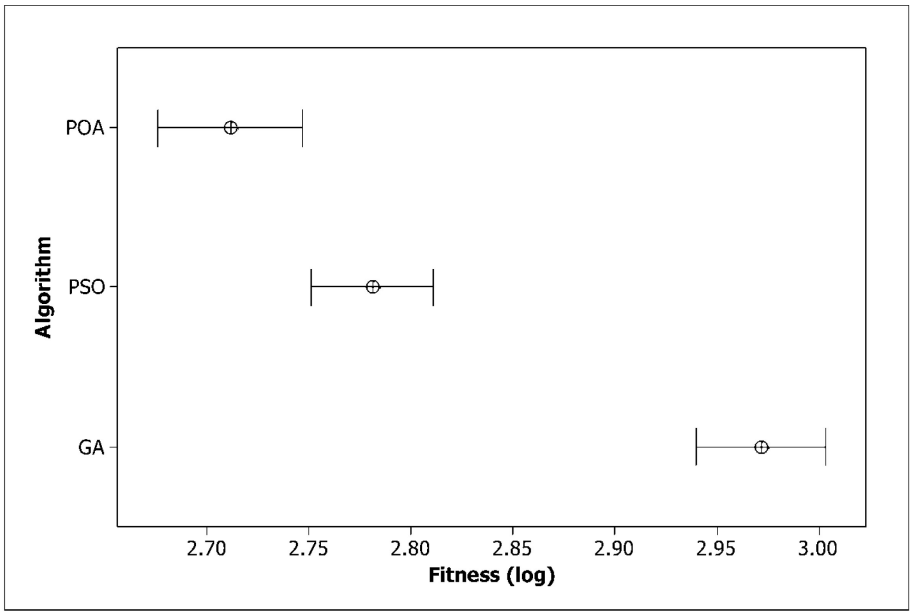


Fig. 10. Confidence intervals of fitness for each algorithm (CI=95% of the mean).

can be used to quickly build learning paths taking already existing resources (repositories of learning units). In terms of curriculum design, sequencers can be useful for gap analysis pointing to resources that are missing in any given curriculum. New learning objects will also be made readily available to the learning community by incorporating them immediately to the learning paths. Management and policy makers can know about the possibility of offering new paths in terms of the resources available in institutional repositories and they could also broaden their catalogs and studies by offering personalized learning paths as an integral part of their programs. In relation to this, an interesting future line of work is data mining because storing the paths and their usage for later mining can point resources in repositories that require closer attention. For example, resources that are critical because are part of many paths or, resources that are missing and can open possibilities to develop new paths with a limited development effort.

Permut-POA can be used to design, build, maintain and test learning paths personalizing and adapting contents to the needs of each particular learner. Personalization is particularly useful in information technology, informatics and computer engineering where there is a myriad of ever evolving technologies and environments. Institutions usually made the decision of what to learn for students but this model may no longer work. Personalization offers the opportunity to put back the student at the center of the learning process offering endless possibilities for adapting contents to her personal needs.

At the methodological level this paper contributes by presenting and applying a systematic method to test and tune algorithms for courseware sequencing. We presented a scalable method to set and fine-tune the parameters of optimizers. Stochastic optimizers usually have a set of parameters that practitioners have to set from scratch. The meaning and potential impact of parameters in the efficiency of algorithms is difficult to understand and usually lies hidden in literature. By setting up a configuration that works for standard problems and also a method to setup and tune the parameters in more challenging scenarios we enable practitioners in engineering education to use optimizers easily to adapt and personalize learning paths.

6.3 Limitations of the study and future work

This study is limited to one test case in engineering education. Additional test cases may be used to determine the applicability of the sequencer in other domains and scenarios. The POA optimizer implements a domain-independent approach based on metadata, so potentially is applicable to any domain of knowledge. As for other scenarios, potential limitations are related with scalability and quality of the metadata descriptions of learning units. Our test case included only 24 learning units (courses). The sequencing problem is potentially an NP (non-polynomial) problem. NP problems are hard to solve because the size of the space of solutions grows exponentially with the size of the input (number of units to sequence). Although population based methods like the ones presented in this work are known for having a good scalability, POA sequencer has to be tested in scenarios that require far more learning objects to be sequenced. Furthermore our test case was designed to simulate a real case and it may not capture the essence or particularities required in other situations. Particularly, we know that the solution space was not very demanding in terms of the number of available solutions, meaning that our test case has many possible valid sequences. The sequencer presented here also has to be tested in more challenging scenarios to claim any kind of generalization. Another potential limitation has to do with the metadata descriptions of learning units. The POA sequencer requires that all learning objects are correctly tagged and that metadata tags are consistent. For instance, a minor difference in the way that a learning output or a prerequisite is defined may result in the impossibility to find a valid sequence or in incorrect paths of learning units. As we have suggested, this can be mitigated using standards for defining the metadata but also by developing standards and common taxonomies of competencies for specific domains.

As for the methods used by the sequencers proposed in this work, we have to point that all the algorithms proposed are stochastic. This means that they can produce different valid sequences given the same initial set of learning units resulting in a potential confusion for the final user (student or course designer). In our opinion this can be mitigated with the right user interface under the assumption that learning sequencers should be used to provide suggestions and recommendations to learners, course builders and curriculum designers. The granularity (size) of learning units can also be a potential source of confusion. Although the proposed sequencer can work with learning objects of any size (e.g. section, module, unit, course), using sets of learning units that have a very dissimilar size (e.g. combining small sections will full courses) can result in inconsistent learning sequences for the final user. We suggest using inputs of learning objects with similar levels of granularity to produce effective results.

It should also be noticed that POA algorithm has too many parameters, so we have just included an

initial analysis concerning parameter tuning that tries to determine which parameters are relevant as well as the most convenient values for the given test case. Experiments carried out suggested that parameters are easy to tune, thus enabling flexible settings of parameters. It has also been observed that some of these parameters are not relevant in terms of performance, e.g., in this particular instance of courseware sequencing we observed that four parameters do not influence performance. In our opinion, further work is required on this field to establish possible dependencies between parameters that further facilitate POA implementation and tuning.

Results also suggested a significant improvement in efficiency when POA is compared to PSO. This may not be so surprising when both metaphors are further inspected because PSO combines local and social exploration, while POA reduces exploration to the social component. POA's capacity to dynamically adjust population size is, in authors' opinion, an important feature that may explain the difference found. We think that further research may also be conducted in order to determine possible equivalences between both optimizers, along with the design of hybrid methods devised to exploit strengths of both optimizers. Comparison with other exact and stochastic methods is also an interesting research that can be undertaken to further explore this parliamentary metaphor. Finally the inclusion of game-like [28-30] and mobile [31–33] approaches can provide a new component to boost participation and motivation.

7. Conclusions

POA's ability to simulate politicians and parties behavior has already proven its efficiency as an optimization method in numerical problems. In this paper, we have presented an adaptation of this social metaphor for solving the problem of courseware sequencing. The courseware sequencing problems has many applications in engineering education for building, testing and maintaining courses, and therefore provides an important practical test-bench for the application of metaheuristics. We have presented a real-world scenario in web engineering of the sequencing problem and used an adapted version of the permut-POA metaheuristic to solve it. Results suggested the potential of POA with respect to other well established optimizers. POA outperformed standard versions of genetic algorithms (GA) and particle swarms (PSO). Improvement was especially significant when POA is compared to a standard genetic algorithm.

Our work can impact on engineering education along the following lines: course builders and curriculum designers could use sequencers to build, maintain and test personalized adapted learning paths. Management can use sequencing to gain further insights of their current resources and how they can be used to leverage new forms of adapted learning. Learners can use sequencers to create personalized paths that can give them recommendations about the set of learning resources to use to acquire the set of competences that they aim for.

Finally we have addressed several limitations in terms of only giving a snapshot limited to one test case in engineering education which raises issues about scalability and generalization. The potential confusion that stochastic methods may cause to users is also a concern. As with other independent-domain sequencing, the effectiveness of our approach depends on the quality of description attached to learning resources (metadata) and their uniformity, especially in terms of size (granularity).

References

1. A. Borji, A new global optimization algorithm inspired by parliamentary political competitions, *Proceedings of the 6th Mexican International Conference on Artificial Intelligence.* pp. 61–71, 2007.
2. A. Borji and M. Hamidi, A new approach to global optimization motivated by parliamentary political competitions. *International Journal of Innovative Computing, Information and Control*. **5**, 2009, pp. 1643–1653.
3. M. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer-Verlag, Berlin, 2012.
4. J. Wilkinson, A matter of life or death: re-engineering competency-based education through the use of a multimedia CD-ROM, *Proceedings of the 2001 IEEE International Conference on Advanced Learning Technologies*. pp. 205–208 (2001).
5. L. de-Marcos, R. Barchino, J. J. Martinez and J. A. Gutierrez, A New Method for Domain Independent Curriculum Sequencing: A Case Study in a Web Engineering Master Program, *International Journal of Engineering Education*, **25**, 2009, pp. 632–645.
6. P. Goodyear, Infrastructure for courseware engineering, in *Automating instructional design: Computer-based development and delivery tools*, pp. 11–31 Springer, Berlin, 1995.
7. P. Brusilovsky, Adaptive and Intelligent Technologies for Web-based Education, *Künstliche Intelligenz, Special Issue on Intelligent Systems and Teleteaching*, **4**, 1999, pp. 19–25.
8. B. El-Falaki, N.-E. El-Faddouli, M. Khalidi-Idrissi and S. Bennani, Individualizing HCI in E-learning Through Assessment Approach. *International Journal of Engineering Education*. **29**, 2014, pp. 650–659.
9. P. Brusilovsky and J. Vassileva, Course sequencing techniques for large-scale web-based education, *Int. J. Cont. Engineering Education and Lifelong Learning*, **12**, 2003, pp. 75–94.
10. J. R. Hilera, S. Oton, A. Ortiz, L. De Marcos, J. J. Martinez, J. A. Gutierrez, J. M. Gutierrez and R. Barchino, Evaluating simple query interface compliance in public repositories, *Proceedings of the 2009 IEEE International Conference on Advanced Learning Technologies*. pp. 311–315 (2009).
11. L. de-Marcos, J.-J. Martinez, J.-A. Gutiérrez, R. Barchino, J.-R. Hilera, S. Oton and J.-M. Gutiérrez, Genetic algorithms for courseware engineering, *International Journal of Innovative Computing, Information and Control*, **7**, 2011, pp. 3981–4004.
12. L. de Marcos, J.-J. Martínez and J.-A. Gutiérrez, Swarm

intelligence in e-learning: a learning object sequencing agent based on competencies, *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, 2008, pp. 17–24.
13. A. Garcia, L. de-Marcos, E. García and J.-A. Gutiérrez, Courseware Sequencing Using Heuristic and Local Search, *Proceedings of the 2010 International Conference on e-Learning, e-Business, Enterprise Information Systems, & e-Government (EEE2010) in the WORLDCOMP'10*. 2010, pp. 70–75.
14. L. de-Marcos, A. García, E. García, J. J. Martínez, J. A. Gutiérrez, R. Barchino, J. M. Gutiérrez, J. R. Hilera and S. Otón, An adaptation of the parliamentary metaheuristic for permutation constraint satisfaction, *Proceedings of the 2010 IEEE Congress on Evolutionary Computation (CEC)*, 2010, pp. 1–8.
15. A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer-Verlag, Berlin, (2003).
16. K. A. de Jong and J. Sarma, On Decentralizing Selection Algorithms, *Proceedings of the 6th international Conference on Genetic Algorithms*, 1995, pp. 17–23.
17. J. E. Baker, Reducing Bias and Inefficiency in the Selection Algorithm, *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and their Application*, 1987, pp. 14–21.
18. X. Hu, R. C. Eberhart and Y. Shi, Swarm intelligence for permutation optimization: a case study of n-queens problem, *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, 2003, pp. 243–246.
19. L. J. Fogel, A. J. Owens and M. J. Walsh, *Artificial Intelligence through Simulated Evolution*. Wiley, Chichester, 1966.
20. J. H. Holland, *Adaptation In Natural and Artificial Systems*, The University of Michigan Press, Michigan, (1975).
21. I. Rechenberg, *Evolution Strategy: Optimizing Technical System based on Biological Evolution Principles*, Fromman-Hozlboog Verlag, Stuttgart, 1973.
22. A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*, Springer-Verlag, Berlin, 2003.
23. S. K. Smit and A. E. Eiben, Comparing parameter tuning methods for evolutionary algorithms, *IEEE Congress on Evolutionary Computation*, 2009, pp. 399–406.
24. C. Chen, Z. Yang, Y. Tan and R. He, Diversity Controlling Genetic Algorithm for Order Acceptance and Scheduling Problem, *Mathematical Problems in Engineering*, 2014, pp. 1–11.
25. R. Poli, Analysis of the publications on the applications of particle swarm optimisation, *Journal of Artificial Evolution and Applications*, 2008, pp. 3.
26. R. Eberhart and J. Kennedy, A new optimizer using particle swarm theory, *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, 1995, pp. 39–43.
27. J. Kennedy and R. Eberhart, Particle swarm optimization, *Proceedings of IEEE international conference on neural networks*, 1995, pp. 1942–1948.
28. K. Kuk and D. Jovanovic, Design and Implementation of CoAeLearn Modules for Personalized Game Based-Learning within Computer Architecture Course, *International Journal of Engineering Education*. **29**, 2013, pp. 620–633.
29. L. de-Marcos, A. Dominguez, J. Saenz-de-Navarrete and C. Pagés, An Empirical Study Comparing Gamification and Social Networking on e-Learning, *Computers and Education*, **75**, 2014, pp. 82–91.
30. A. Domínguez, J. Saenz-de-Navarrete, L. de-Marcos, L. Fernández-Sanz, C. Pagés and J. J. Martínez-Herráiz, Gamifying learning experiences: Practical implications and outcomes, *Computers and Education*, **63**, 2013, pp. 380–392.
31. A. Garcia-Cabot, L. de-Marcos and E. Garcia-Lopez, An empirical study on m-learning adaptation: Learning performance and learning contexts, *Computers and Education*, **82**, 2015, pp. 450–459.
32. A. García-Cabot, E. García-López, L. de-Marcos, L. Fernández and J.-M. Gutiérrez-Martínez, Adapting Learning Content to User Competences, Context and Mobile Device using a Multi-Agent System: Case Studies, *International Journal of Engineering Education*, **30**, 2014, pp. 937–949.
33. L. de-Marcos, R. Barchino, L. Jiménez, J. R. Hilera, J. J. Martinez, J. A. Gutierrez, J. M. Gutierrez and S. Otón, Using M-Learning on Nursing Courses to Improve Learning, *CIN: Computers, Informatics, Nursing*, **29**, 2011, pp. 311–317.

Luis de-Marcos has a BSc (2001) and an MSc (2005) in Computer Science from the University of Alcalá, where he also completed his PhD in the Information, Documentation and Knowledge program in 2009. He is currently working in the Computer Science Department as an assistant professor. He has over 90 publications in relevant conferences and journals. He has also been a research fellow at the Lund University (Sweden), the University of Reading (UK) and the Monterrey Institute of Technology (México).

Antonio Garcia-Cabot has an MSc (2009) in Computer Science from the University of Alcala, where he now occupies a researcher position in the Computer Science Department. He finished his PhD (2013) in Information and Knowledge Engineering. His research interests include intelligent agents, adaptable and adaptive systems, mobile devices and usability. He has more than 40 publications in relevant conferences and journals. He participates in different national and European research projects.

Eva Garcia-Lopez has a BSc (2007) and an MSc (2009) in Computer Science from the University of Alcala, where she is now a researcher in the Computer Science Department. She finished her PhD (2013) in Information and Knowledge Engineering. Her research interests include usability and mobile devices. She participates in different research projects and she has publications in relevant conferences and journals.

José Amelio Medina has a Telecommunications Engineering degree from the University of Alcalá, an Official Master in Company Management and PhD on Administration and Business Management from Rey Juan Carlos University, being also graduated in Marketing and Market Research. Currently, he is an Assistant Professor in the Computer Science Department of the University of Alcala. Previously, he acted as a Technical Management (Systems and Information Technology) in the Health Care Service of Castille-Mancha, Computer Technician in ADAC Association, and as Head of Quality, Environment and Computer Services in Tecnivial, SA Company.