Journal of Universal Computer Science, vol. 22, no. 10 (2016), 1298-1318
submitted: 23/12/15, accepted: 30/9/16, appeared: 1/10/16 © J.UCS

A Steady-state Evolutionary Algorithm for Building Collaborative Learning Teams in Educational Environments Considering the Understanding Levels and Interest Levels of the Students

Virginia Yannibelli
(ISISTAN (CONICET-UNCPBA), Campus Universitario, Tandil, Argentina
virginia.yannibelli@isistan.unicen.edu.ar)

Marcelo Armentano
(ISISTAN (CONICET-UNCPBA), Campus Universitario, Tandil, Argentina
marcelo.armentano@isistan.unicen.edu.ar)

Franco Berdun
(ISISTAN (CONICET-UNCPBA), Campus Universitario, Tandil, Argentina
franco.berdun@isistan.unicen.edu.ar)

Analía Amandi
(ISISTAN (CONICET-UNCPBA), Campus Universitario, Tandil, Argentina
analia.amandi@isistan.unicen.edu.ar)

Abstract: Collaborative learning team building is a fundamental, difficult and time-consuming task in educational environments. In this paper, we address a collaborative learning team building problem that considers two valuable grouping criteria usually considered by teachers. One of these criteria considers the understanding levels of the students with respect of the topics of a given course, and is based on building well-balanced teams in terms of the understanding levels of their members. The other criterion considers the interest levels of the students with respect of the topics of a given course, and is based on building well-balanced teams in terms of the interest levels of their members. The problem addressed has been recognised as an NP-Hard optimization problem. To solve the problem, we propose a steady-state evolutionary algorithm. This algorithm aims to organize the students taking a given course into teams in such a way that the two grouping criteria of the problem are optimized. The performance of the algorithm is evaluated on nine problem instances with different levels of complexity, and is compared with that of the only algorithm previously proposed for solving the addressed problem. The obtained results show that the steady-state evolutionary algorithm significantly outperforms the previous algorithm.

Keywords: Collaborative Learning, Collaborative Learning Team Building, Understanding Levels, Interest Levels, Evolutionary Algorithms, Steady-State Evolutionary Algorithms
Categories: G.1.6, I.2.8, J.4, K.3, K.3.1, L.3, L.3.6, L.6.2

1 Introduction

Collaborative learning is an instructional approach usually used in educational environments in order to supplement and enrich the individual learning of the students

[Barkley et al., 2005; Michaelsen et al., 2004]. This approach requires organizing the students into collaborative learning teams. Then, the students of each collaborative learning team must work together to achieve shared learning goals. The collaborative learning teams must be built in such a way that the students can acquire new knowledge and skills through the interaction with their peers, improving their individual learning. Thus, the building of collaborative learning teams from the students is a fundamental task in collaborative learning.

To build collaborative learning teams from the students, teachers must utilize some grouping criterion (i.e., criterion to form collaborative learning teams). The grouping criterion is really important because of the way in which a team is made up affects the learning level and the social behavior of the students belonging to the team as well as the performance of the team [Barkley et al., 2005; Michaelsen et al., 2004]. Besides, the way in which the grouping criterion is applied (i.e., either manually or automatically) is important since many known grouping criteria require a considerable amount of knowledge, time and effort to be manually applied [Cruz and Isotani, 2014]. In these cases, it is possible to considerably reduce the workload of teachers and optimize the collaborative learning team building through automation.

Different works in the literature have described and addressed the problem of building collaborative learning teams automatically from the students [Alberola et al., 2016; Cruz and Isotani, 2014]. These works significantly differ in relation to several aspects including the students' characteristics analyzed, the grouping criteria considered, and the algorithms utilized. In this respect, to the best of our knowledge, only few works have considered grouping criteria that both are usually considered in real-world classrooms by teachers and have been successfully evaluated in different kinds of educational environments

In [Lin et al., 2010], the authors describe the problem of building collaborative learning teams automatically from the students taking a given course. As part of the problem, the authors consider two grouping criteria that must be simultaneously satisfied. One of these criteria considers the understanding levels of the students in respect of each of the topics of the course, and is based on building well-balanced teams regarding the understanding levels of their members in respect of each topic. The other criterion considers the interest levels of the students in respect of each of the topics of the course, and is based on building well-balanced teams regarding the interest levels of their members in respect of each topic. These two grouping criteria are usually considered by teachers in real-world classrooms [Saleh and Kim, 2009]. Moreover, different works in the literature [Michaelsen et al., 2004; Yang, 2006; Saleh and Kim, 2009; Nielsen et al., 2009] indicate that collaborative learning team building based on these two criteria leads to good discussions and interactions during the learning process, improves the social behavior of the students, enhances the learning process of the students, and impacts positively on the learning level of the students as well as on the performance of the teams. Thus, it is considered that the collaborative learning team building problem described in [Lin et al., 2010] is really valuable in the context of collaborative learning.

The collaborative learning team building problem described in [Lin et al., 2010] is an NP-Hard optimization problem. Because of this, as reported in [Lin et al., 2010], exhaustive search algorithms only can solve small instances of the problem in a reasonable period of time. Thus, heuristic search algorithms are required to solve the problem. In this respect, only one heuristic search algorithm has been proposed in the literature to solve the problem. We refer to the particle swarm optimization algorithm proposed in [Lin et al., 2010].

In this paper, we address the collaborative learning team building problem described in [Lin et al., 2010] with the aim of proposing a better heuristic search algorithm to solve it. In this regard, we propose a steady-state evolutionary algorithm. Given a course that aims to teach a number of topics, and a number of students taking the course who must be organized into a given number of collaborative learning teams, the algorithm explores different solutions to organize the students into collaborative learning teams, with the aim of finding the solutions that optimize the two grouping criteria considered as part of the problem. The explored solutions are evaluated with respect of the two grouping criteria. To perform that evaluation, the algorithm is based on knowledge of the understanding levels and interest levels of the students with respect of each of the topics of the course.

We propose a steady-state evolutionary algorithm because of the following reasons. Evolutionary algorithms have been proved to be effective and efficient in the resolution of a wide variety of NP-Hard optimization problems [Eiben and Smith, 2015; Deb, 2009] and, in particular, in the resolution of collaborative learning team building problems [Cruz and Isotani, 2014]. Besides, evolutionary algorithms have been shown to be more effective than other heuristic search algorithms (e.g., particle swarm optimization algorithms) in the resolution of different NP-Hard optimization problems [Saishanmuga Raja and Rajagopalan, 2014; Kachitvichyanukul, 2012]. Thus, we consider that the proposed steady-state evolutionary algorithm could outperform the heuristic search algorithm previously proposed in the literature for solving the addressed problem.

The remainder of the paper is organized as follows. In Section 2, we present a brief review of published works that address the problem of building collaborative learning teams automatically. In Section 3, we describe the collaborative learning team building problem addressed in this paper. In Section 4, we present the steady-state evolutionary algorithm proposed to solve the addressed problem. In Section 5, we present the computational experiments developed to evaluate the performance of the steady-state evolutionary algorithm, and an analysis of the results obtained. Finally, in Section 6, we present the conclusions of the present work.

2 Related Works

In the literature, different works have described and addressed the problem of building collaborative learning teams automatically [Cruz and Isotani, 2014]. These works significantly differ regarding several aspects including the students' characteristics analyzed, the grouping criteria considered, and the algorithms and measures used. In this section, we review related works reported in the literature, focusing the attention on analyzing the aspects above-mentioned.

Some works in the literature propose approaches for collaborative learning team building considering grouping criteria based on the learning styles, the team roles, or the thinking styles of the students. In [Christodoulopoulos and Papanikolaou, 2007], a web-based tool is presented which considers the following students' characteristics: the dimensions of the Felder-Silverman learning style model, the dimensions of the

Honey-Mumford learning style model, and the knowledge level for the current lesson. This tool provides both a Fuzzy C-Means algorithm for building intra-homogeneous teams regarding each of the students' characteristics, and a standard random selection algorithm for building intra-heterogeneous teams regarding each of the students' characteristics. The tool allows the teacher to select the algorithm to be used for building the teams, as well as the students' characteristics to be considered. The teacher can select only up to three students' characteristics.

Alberola et al. (2016) propose a tool based on the Belbin's team role model. This tool aims to build well-balanced teams in respect of the team roles of their student members. In this case, the collaborative learning team building problem is modeled as a coalition structure generation problem and is solved by means a linear programming method. This method uses a measure to estimate the balance level of the possible teams. Although this measure considers the main balance condition defined by Belbin (i.e., one student per role), the measure does not consider all the unbalance conditions defined by Belbin (i.e., missing roles are not considered). The team roles of each student are estimated only from the feedback given by the other students, by using Bayesian learning. Although this is meant to avoid the drawbacks inherent to the use of the Team Role Self-Perception Inventory, the estimation of the students' roles could be negatively affected by biased feedback.

In [Yannibelli and Amandi, 2012a, 2012b, 2013], the authors present different hybrid evolutionary algorithms that consider the Belbin's team role model. These algorithms have the aim of building well-balanced teams in respect of the team roles of their student members. In this sense, the algorithms use a measure to estimate the balance level of the possible teams. In contrast with the work presented in [Alberola et al., 2016], this measure considers all the balance conditions and the unbalance conditions defined by Belbin.

Ounnas et al. (2009) propose a framework that uses an ontology to describe some student's characteristics including Honey-Mumford learning styles and Belbin's team roles. The framework provides a list with only a few grouping criteria. Each criterion refers to one student's characteristic described in the ontology (e.g., learning style), and indicates a constraint about such characteristic which should be satisfied (e.g., homogeneity or heterogeneity). In this framework, the collaborative learning team building problem is modeled as a constraint satisfaction problem, and is solved by a DLV constraint satisfaction solver. The weak constraints of the problem refer to the grouping criteria selected by the teacher from the provided list, and the optimization objective of the problem is to find the set of teams that minimizes the number of violated weak constraints.

Wang et al. (2007) present a collaborative learning team building system that considers the thinking styles of the students, according to the Sternberg's thinking styles model. The thinking styles of the students are collected via a thinking style questionnaire. The system uses a non-elitist genetic algorithm with the aim of building intra-heterogeneous and inter-homogeneous teams regarding the thinking styles of their student members. However, this genetic algorithm analyzes the heterogeneity of the possible teams only in respect of a few thinking styles of the Sternberg's thinking styles model.

In the above-mentioned works, the authors consider grouping criteria based on the Honey-Mumford or Felder-Silverman learning styles, the Belbin's team roles, or the Sternberg's thinking styles of the students. However, such grouping criteria are not usually considered in real-world classrooms by teachers [Alberola et al., 2016].

A number of works in the literature propose approaches for collaborative learning team building considering grouping criteria designed for specific courses. In [Graf and Bekele, 2006], the authors propose an ant colony optimization algorithm that considers six students' characteristics (i.e., group work attitude, interest for the subject, achievement motivation, self-confidence, shyness, fluency in the language of instruction, and level of performance in the subject). This algorithm aims to build intra-heterogeneous teams regarding the characteristics of their student members. In this sense, a measure is proposed to estimate the heterogeneity level of the possible teams. However, this measure does not analyze the heterogeneity of the teams in respect of each one of the considered characteristics, showing significant limitations for estimating the heterogeneity level of the teams.

Meyer (2009) presents a web-based tool that considers the preferences of the students in respect of the projects of the course. Each student is asked provide preference levels for the projects of the course. In this tool, the collaborative learning team building problem is modeled as a constraint satisfaction problem and is solved by a linear programming solver. The optimization objective of this problem implies building intra-homogeneous collaborative learning teams with respect to the student preferences, maximizing the student satisfaction regarding the satisfied preferences.

Zhamri Che Ani et al. (2010) propose a non-elitist genetic algorithm for building teams in the context of software programming university courses. This algorithm considers the programming skills of the students, and aims to build heterogeneous teams in respect of the programming skills of their student members.

Only a few works in the literature propose approaches for collaborative learning team building considering an unlimited number of students' characteristics. Moreno et al. (2012) propose a non-elitist genetic algorithm that considers an arbitrary number of students' characteristics. The algorithm aims to build inter-homogeneous and intra-heterogeneous collaborative learning teams in respect of the students' characteristics. In this regard, the algorithm uses a generic measure to estimate the homogeneity level of the possible sets of collaborative learning teams. However, this measure has significant limitations for developing correctly such estimations. These limitations are mainly because of the additive nature of the measure.

Isotani et al. (2009) propose a collaborative learning team building method driven by learning theories, although these authors mention that the use of learning theories to support collaborative learning is open for criticism. In this case, learning theories are considered as guidelines to support collaborative learning team building as well as collaborative learning activity design. To facilitate the use of such theories by the method, an ontology is proposed that represents knowledge extracted from existing learning theories. Specifically, this ontology represents concepts, and relations among concepts, that are considered relevant to support collaborative learning team building and collaborative learning activity design (e.g, individual learning goal, group learning goal, role, learning strategy, interaction pattern, and students' stage of knowledge/skill). The ontology is used by the method to both collaborative learning team building and collaborative learning activity design. The method only has been partially evaluated on one small case study.

In [Lin et al., 2010], a collaborative learning team building problem is modeled mathematically, which considers two grouping criteria that must be simultaneously optimized. One of these criteria considers the understanding levels of the students in respect of each of the topics of the course, and is based on building well-balanced teams regarding the understanding levels of their members in respect of each topic. The other criterion considers the interest levels of the students in respect of each of the topics of the course, and is based on building well-balanced teams regarding the interest levels of their members in respect of each topic. These two grouping criteria are usually considered by teachers in real-world classrooms [Saleh and Kim, 2009]. Besides, different works in the literature [Michaelsen et al., 2004; Yang, 2006; Saleh and Kim, 2009; Nielsen et al., 2009] indicate that collaborative learning team building based on these two grouping criteria leads to good discussions and interactions during the learning process, improves the social behavior of the students, enhances the learning process of the students, and impacts positively on the learning level of the students as well as on the performance of the teams. However, the manual building of collaborative learning teams considering simultaneously these two criteria is a really complex, costly, and time-consuming task for teachers. For these reasons, it is really valuable to propose an effective and efficient algorithm for automatically solving the problem modeled in [Lin et al., 2010], with the aim of assisting teachers in building collaborative learning teams based on the two mentioned criteria.

3 Problem Description

In this paper, we address the collaborative learning team building problem presented in [Lin et al., 2010]. A description of this problem is presented below.

Suppose that a course aims to teach k topics, and n students taken the course. The teacher of the course must organize the n students into g teams. Each team must be made up of a number of member students, and each student can only belong to one team. Regarding team size, students must be organized in such a way that the teams have a similar number of students each. Specifically, the difference between the size of a team and the size of the other teams must not exceed one. The values of the terms k, n and g are known.

As regards the students, it is considered that they have different understanding levels and different interest levels in relation to each of the k topics. The understanding level of a student s in relation to a topic l is notated as U_{sl}, and the interest level of a student s in relation to a topic l is notated as L_{sl}, considering $1 \leq s \leq n$ and $1 \leq l \leq k$. The mentioned terms U_{sl} and L_{sl} take a real value over the range [0, 1]. The values of the terms U_{sl} and L_{sl} inherent to each student are known. As described in [Lin et al., 2010], the values of the terms U_{sl} inherent to each student can be obtained through a pre-test specially designed to cover the k topics. In addition, the values of the terms L_{sl} inherent to each student can be obtained by analyzing available information about the participation of the students in already developed learning activities (e.g., accessed learning material, topics discussed, and answers to interest questionnaires about the topics).

As part of the problem, teams must be made up in such a way that two grouping criteria are reached. One of these grouping criteria is to minimize the difference in understanding level for each topic among the teams. This criterion aims to build well-balanced teams regarding the understanding levels of their student members in respect of each topic. The other grouping criterion is to minimize the difference in interest level for each topic among the teams. This criterion aims to build well-balanced teams regarding the interest levels of their student members in respect of each topic. These grouping criteria require analyzing the understanding levels and the interest levels of the formed teams in relation to each topic. In this respect, the understanding level of a team with respect of a given topic depends on the understanding levels of the students belonging to the team with respect of the topic. The interest level of a team with respect of a given topic depends on the interest levels of the students belonging to the team with respect of the topic.

The grouping criteria considered as part of the problem are modeled by Equations (1), (2), (3) and (4).

Equation (1) minimizes the difference in understanding level for each topic among the *g* teams defined from the *n* students, and minimizes the difference in interest level for each topic among the *g* teams defined from the *n* students. In other words, the objective of this equation is to find a solution (i.e., a set of *g* teams) that minimizes the difference in understanding level for each topic among the *g* teams and minimizes the difference in interest level for each topic among the *g* teams. This is the optimal solution to the addressed problem. In Equation (1), set *C* contains all the sets of *g* teams that may be defined from the *n* students. The term *G* represents a set of *g* teams belonging to *C*. The term $Z(G)$ represents the difference in understanding level and interest level for the *k* topics among the *g* teams belonging to set *G*. Equation (1) uses Equations (2), (3) and (4) to establish the mentioned difference among the *g* teams belonging to set *G*. The term $Z(G)$ takes a real value over the range [0, 3]. In the case of a *G* set of *g* teams with no difference in understanding level and interest level for the *k* topics, the value of the term $Z(G)$ is equal to 0, considering 0 as the best possible value for $Z(G)$.

Equation (2) establishes the average maximal difference in understanding level for the *k* topics among the *g* teams belonging to set *G*. To establish this average maximal difference, the equation analyzes the maximal difference in understanding level for each topic among the *g* teams belonging to set *G*. In this equation, the term w_l represents the weight assigned to the topic *l* and takes a real value over the range [0, 1]. The term U_{xjl} represents the understanding level of the student *x* of the team *j* with respect of the topic *l*, and takes a real value over the range [0, 1]. The term *c* represents the maximal size of the *g* teams belonging to set *G*. Note that the term $f(G)$ takes a real value over the range [0, 1].

Equation (3) establishes the maximal difference in understanding level for the *k* topics among the *g* teams belonging to set *G*. To determine this maximal difference, the equation analyzes the maximal difference in understanding level for each topic among the *g* teams belonging to set *G*. Note that the term $C_1(G)$ takes a real value over the range [0, 1]. As mentioned in [Lin et al., 2010], Equation (3) complements Equation (2) in some particular situations. In this respect, when there is a slight difference in understanding level for most of the *k* topics among the *g* teams, and there is a significant difference in understanding level for a few of the *k* topics among the *g* teams, then the value of $f(G)$ in Equation (2) remains small. This is because of Equation (2) averages the differences in understanding level for all the *k* topics among the *g* teams. Thus, the value of $f(G)$ in Equation (2) hides the more significant difference in understanding level for the k topics among the g teams. As a result, the mentioned difference can not be appropriately minimized when only Equation (2) is used in the explained situations. To avoid this happen, Equation (2) is used together with Equation (3). Equation (3) establishes the more significant difference in understanding level for the k topics between the g teams. As a result, the mentioned difference can be minimized.

Equation (4) establishes the maximal gap in interest level for the g teams belonging to set G with respect of the k topics. To establish this maximal gap, the equation analyzes the gap in interest level for the g teams with respect of each topic. In this equation, the term L_{xjl} represents the interest level of the student x of the team j with respect of the topic l, and takes a real value over the range [0, 1]. Note that the term $C_2(G)$ takes a real value over the range [0, 1].

For a more detailed discussion of Equations (1), (2), (3) and (4), readers are referred to the work [Lin et al., 2010], which has introduced these equations.

$$\min_{\forall G \in C} \left(Z(G) = f(G) + C_1(G) + C_2(G) \right) \tag{1}$$

$$f(G) = \frac{\sum_{l=1}^{k} \left\{ \max_{1 \le j \le g} \left\{ \sum_{x=1}^{c} \left(w_l U_{xjl} \right) \right\} - \min_{1 \le j \le g} \left\{ \sum_{x=1}^{c} \left(w_l U_{xjl} \right) \right\} \right\}}{ck} \tag{2}$$

$$C_1(G) = \frac{\max_{1 \le l \le k} \left\{ \max_{1 \le j \le g} \left\{ \sum_{x=1}^{c} \left(w_l U_{xjl} \right) \right\} - \min_{1 \le j \le g} \left\{ \sum_{x=1}^{c} \left(w_l U_{xjl} \right) \right\} \right\}}{c} \tag{3}$$

$$C_2(G) = \max_{1 \le l \le k} \left\{ 1 - \frac{\sum_{j=1}^{g} \min \left\{ 1, \sum_{x=1}^{c} L_{xjl} \right\}}{g} \right\} \tag{4}$$

4 A Steady-State Evolutionary Algorithm

To solve the addressed problem, we propose a steady-state evolutionary algorithm. The steady-state evolutionary algorithms are population-based stochastic search and optimization algorithms inspired by the theory of evolution of species proposed by Darwin in 1859 [Eiben and Smith, 2015; Deb, 2009]. These algorithms search the solution space of a given problem in order to find the optimal solutions, applying the Darwinian principles of selection, crossover and mutation.

4.1 General Behavior of the Steady-State Evolutionary Algorithm

The general behavior of the steady-state evolutionary algorithm proposed here is shown in Fig. 1 and is described as follows. Given a course that aims to teach *k* topics, and given the *n* students taking the course who shall be organized into *g* collaborative learning teams, the evolutionary algorithm starts from an initial population of feasible solutions. Each solution of this initial population encodes a feasible set of *g* teams which may be defined from the *n* students taking the course. Then, each solution of the population is decoded (i.e., the set of *g* teams inherent to the solution is built), and evaluated with respect of the two grouping criteria of the problem by a fitness function. As mentioned in Section 3, one of these grouping criteria is to minimize the difference in understanding level for the *k* topics among the *g* teams, and the other grouping criterion is to minimize the difference in interest level for the *k* topics among the *g* teams. Therefore, considering a given solution, the fitness function evaluates the understanding levels and the interest levels of the *g* teams represented by the solution in relation to each of the *k* topics. To perform that evaluation, the function is based on knowledge of the understanding levels and interest levels of the *n* students with respect of the *k* topics of the course.

Steady-state evolutionary algorithm

inputs: population_size, number_generations, P_c, P_m
outputs: best solution from the last generation or population

```
procedure:
 1: population = generate_initial_population(population_size);
 2: generation = 1;
 3: while (generation ≤ number_generations) do
 4:     mating_pool = parent_selection_process(population);
 5:     offprings = crossover_process(mating_pool, $P_c$);
 6:     mutation_process(offsprings, $P_m$);
 7:     population = steady_state_survival_selection(population,offprings);
 8:     generation = generation + 1;
 9: end while
10: solution = best_solution_from(population);
11: return solution;
```

Figure 1: Description of the steady-state evolutionary algorithm.

After the solutions of the population are evaluated, a parent selection process is used in order to determine which solutions of the population will compose the mating pool. In this respect, the solutions with the best fitness values will have more chances of being selected. After the mating pool is composed, the solutions in the mating pool are paired, and a crossover process is applied to each pair of solutions with a probability P_c in order to generate new feasible ones. Then, a mutation process is applied to each solution generated by the crossover process with a probability P_m. The mutation process is applied in order to introduce diversity in the solutions generated by the crossover process. Finally, a fitness-based survival selection process for steady-state evolutionary algorithms is used. This process is used in order to determine which solutions from the solutions in the population and the solutions generated from the mating pool will compose the new population.

This process is repeated until a predetermined number of generations is reached. After this happens, the algorithm provides the solution with the best fitness value within the last population or generation as a solution to the addressed problem.

4.2 Components of the Steady-State Evolutionary Algorithm

In the next sections, the main components of the steady-state evolutionary algorithm are described. These components are the encoding and decoding of solutions, the fitness function, and the parent selection, crossover, mutation and survival selection processes.

4.2.1 Encoding and Decoding of Solutions

In the population of the steady-state evolutionary algorithm, each solution represents a feasible set of g teams which may be defined from the n students taking the course. To encode the solutions of the population, we used the encoding described in [Yannibelli and Amandi, 2012a, 2012b]. By using this encoding, each solution is encoded as a list with as many positions as students taking the course (i.e., a list with a length equal to n). Each position on this list contains a different student (i.e., repeated students are not admitted on the list). Besides, each student may be in any position on the list. Thus, the list is a permutation of the n students.

In order to build a set of g teams from the above-described list, we used the decoding process proposed in [Yannibelli and Amandi, 2012a, 2012b]. This process builds a set of g teams from the list taking into account the two restrictions considered as part of the addressed problem. The first restriction is that each student may belong to only one team. The second restriction is that the difference between the size of a team and the size of the rest of the teams must not exceed one. The decoding process works as follows.

In the decoding process, the size of the teams depends on the relationship between the values n and g. Thus, the process starts by calculating the value of the term $z = (n/g)$. When z is an integer, then the list is divided into g equal segments, each of which has a size equal to z and represents to a different team. Thus, g teams with the same size are built.

When z is not an integer (i.e., z is a real number), g teams with the same size can not be built. Besides, the process considers that the difference between the sizes of any two teams must not exceed one. Thus, the process builds g teams which do not have the same size, but which respect the restriction mentioned above. Specifically, the process divides the list into g segments: the first g_1 segments have a size equal to ((integer part of z) + 1) and the remaining segments have a size equal to (integer part of z), considering $g_1 = (n - ((\text{integer part of } z) \times g))$. Each segment represents to a different team. Thus, the process builds g_1 teams with a size equal to ((integer part of z) + 1) and $(g - g_1)$ teams with a size equal to (integer part of z).

In relation to the behavior of the above-described decoding process, note that when this process is applied, only one set of g teams can be built from a given encoded solution, but different encoded solutions could be transformed in the same set of g teams.

In relation to the generation of the encoded solutions of the initial population, we used the random-based generation process described in [Yannibelli and Amandi, 2012a, 2012b]. By using this process, a diverse initial population is obtained. This is meant to avoid the early stagnation of the search developed by the steady-state evolutionary algorithm.

4.2.2 Fitness Function

This function is used by the steady-state evolutionary algorithm to determine the fitness values of the encoded solutions. The fitness value of an encoded solution represents the quality of the related set of g teams with respect of the two grouping criteria of the addressed problem. As was mentioned in Section 3, one of these grouping criteria is to minimize the difference in understanding level for the k topics among the g teams, and the other grouping criterion is to minimize the difference in interest level for the k topics among the g teams. Thus, the fitness function evaluates a given encoded solution in relation to each one of the two mentioned grouping criteria and then defines a scalar fitness value for the solution based on the results obtained by the evaluations.

The detailed behavior of the fitness function is described as follows. Considering a given encoded solution, the function decodes the G set of g teams related to the solution by using the decoding process described in Section 4.2.1. Then, the function calculates the value of the term $Z(G)$ corresponding to G (Equations (1), (2), (3) and (4)). This value represents the difference in understanding level and interest level for the k topics among the g teams composing the G set, and therefore, determines the fitness level of the encoded solution. In the case of a G set of g teams with no difference in understanding level and interest level for the k topics, the value of the term $Z(G)$ is equal to 0, considering 0 as the best possible fitness level.

To calculate the value of the term $Z(G)$, the fitness function utilizes the values of the terms U_{sl} and L_{sl} inherent to G (Equations (2), (3) and (4)). As was mentioned in Section 3, the values of the terms U_{sl} and L_{sl} inherent to each student s are known. In this respect, a knowledge base contains the values of the terms U_{sl} and L_{sl} inherent to each of the n students taking the course. Then, the fitness function queries the knowledge base to obtain the values of the terms U_{sl} and L_{sl} inherent to each student s.

4.2.3 Parent Selection Process

In the steady-state evolutionary algorithm, the parent selection process is utilized in order to determine which solutions of the current population will compose the mating pool. This process is very relevant because of the solutions in the mating pool, usually called parent solutions, will be used by the crossover process to generate new solutions, usually called offspring solutions.

To develop the parent selection, we applied the process called deterministic tournament selection with replacement [Eiben and Smith, 2015]. This process is a variant of the traditional tournament selection process. By using the process deterministic tournament selection with replacement, the solutions with the best fitness values within the current population will have more chances of being incorporated in the mating pool.

The process deterministic tournament selection with replacement works as follows. A number of t solutions are randomly selected from the current population, where $2 \leq t \leq P$ and P is the population size. The t selected solutions compete for being incorporated in the mating pool. The better one (i.e., the solution with the best fitness value) is incorporated into the mating pool. Then, the t solutions are returned to the population. The described operation is repeated until a number P of solutions is incorporated in the mating pool.

4.2.4 Crossover Process

Once the mating pool is composed, the solutions in the mating pool are paired considering the order in which they where incorporated in the mating pool. Subsequently, a crossover process is applied to each of these pairs of solutions with a probability P_c to generate new solutions. Specifically, the crossover process applied to a pair of solutions, called parent solutions, combines the characteristics of these solutions and generates two new solutions, called offspring solutions. Therefore, the crossover process has the possibility of combining the best characteristics of the parent solutions so that new, better solutions can be generated [Eiben and Smith, 2015].

In relation to the crossover process applied by the steady-state evolutionary algorithm, we considered a feasible process for solutions encoded as permutations of n elements. This is because of the solutions in the algorithm are encoded as permutations of n elements (i.e., permutations of n students). Specifically, we considered a crossover process called partially mapped crossover [Eiben and Smith, 2015]. This process is one of the most applied for permutations in the literature [Eiben and Smith, 2015].

The partially mapped crossover process works as follows. Given two solutions parent 1 and parent 2 (i.e., two permutations of the n students), the process creates copies offspring 1 and offspring 2 of parent 1 and parent 2, respectively. Then, the process defines two random crossover points $c1$ and $c2$, considering $1 < c1 < c2 < n$.

To generate the first offspring from the given solutions parent 1 and parent 2, the process modifies offspring 1 by the following procedure. For positions $j = c1, \ldots, c2$, the process observes the student on the position j of parent 2, and then searches for the position j' of this student in offspring 1. Then, the process exchanges the students on the positions j and j' of offspring 1.

To generate the second offspring from the given solutions parent 1 and parent 2, the process modifies offspring 2 by the following procedure. For positions $j = c1,\ldots, c2$, the process observes the student on the position j of parent 1, and then searches for the position j' of this student in offspring 2. Then, the process exchanges the students on the positions j and j' of offspring 2.

The above-described procedures generate two new feasible solutions offspring 1 and offspring 2 (i.e., two new feasible permutations of the n students) from the given solutions parent 1 and parent 2.

4.2.5 Mutation Process

The previously described crossover process generates a new set of solutions from the mating pool. Subsequently, a mutation process is applied to each solution of this new set, with a probability P_m. This is meant to randomly modify one or more characteristics of some of the solutions generated by the crossover process, and thus, to introduce genetic diversity in these solutions [Eiben and Smith, 2015].

In relation to the mutation process applied by the steady-state evolutionary algorithm, we considered a feasible process for solutions encoded as permutations of n elements. This is because of the solutions in the algorithm are encoded as permutations of n elements (i.e., permutations of n students). Specifically, we considered a mutation process called exchange mutation [Eiben and Smith, 2015]. This process is one of the most applied for permutations in the literature [Eiben and Smith, 2015].

The exchange mutation process works as follows. Given a solution (i.e., a permutation of n students), the process randomly selects two positions $p1$ and $p2$ on the solution, considering $1 \leq p1 < p2 \leq n$. Then, the process exchanges the students on the positions $p1$ and $p2$ of the solution. Thus, the described mutation process always generates a new feasible solution (i.e., a new feasible permutation of the n students) from the given solution.

4.2.6 Survival Selection Process

In the steady-state evolutionary algorithm, the survival selection process is used in order to determine which solutions from the solutions in the current population and the solutions generated from the mating pool will compose the new population.

To develop the survival selection, we applied a fitness-based survival selection process for steady-state evolutionary algorithms [Eiben and Smith, 2015]. By using this process, the best solutions found by the evolutionary algorithm are preserved.

The mentioned fitness-based survival selection process works as follows. First, the process selects the best $(P - \lambda)$ solutions from the current population, where P is the population size, and λ is a parameter that takes an integer value over the range $[1, P - 1]$. Then, the process selects the best λ solutions from the solutions generated from the mating pool by the crossover and mutation processes. Finally, the process uses the P selected solutions to compose the new population.

5 Computational Experiments to Evaluate the Steady-State Evolutionary Algorithm

In this section, we describe the computational experiments developed to evaluate the performance of the steady-state evolutionary algorithm. After that, we present and analyze the results obtained by the experiments. Finally, we compare the performance of the steady-state evolutionary algorithm with that of the particle swarm optimization algorithm presented in [Lin et al., 2010] for solving the addressed problem. To the best of our knowledge, the algorithm presented in [Lin et al., 2010] is the only algorithm previously proposed in the literature for solving the addressed problem.

To develop the computational experiments, we designed nine data sets. Table 1 shows the main characteristics of each data set. Each data set contains a list of k topics and a list of n students. The k topics have the same weight (i.e., the k topics have a weight equal to 1). For each data set, we established a g number of teams to be built from the n students. Note that the size of the g teams is equal to 6 members. In the literature, this size is considered one of the optimal sizes for collaborative learning teams [Barkley et al., 2005; Michaelsen et al., 2004].

Moreover, for each student of each of the nine data sets, we defined a specific understanding level and a specific interest level with respect of each of the k topics. The understanding levels and interest levels of the n students with respect of the k topics take a real value on the range [0, 1], as mentioned in Section 3. Specifically, in each data set, for each of the k topics, g students have an understanding level equal to 1 and the remaining students have an understanding level equal to 0. Moreover, in each data set, for each of the k topics, g students have an interest level equal to 1 and the remaining students have an interest level equal to 0.

Thus, in each data set, each of the k topics is understood by g students of the data set, and each of the k topics is interesting for g students of the data set. In this way, from the n students of the data set, it is possible to build at least one set of g teams with no difference in understanding level and interest level with respect of each of the k topics. In other words, it is possible to build at least one set of g teams in which each of the k topics is understood by one team member and is interesting for one team member. According to the fitness function described in Section 4.2.2, a set of g teams with no difference in understanding level and interest level regarding each of the k topics has a fitness level equal to 0. This fitness level is the best possible fitness level.

Based on the mentioned, we may state that there is at least one set of g teams with the best possible fitness level for each data set. Considering that a set of g teams with the best possible fitness level outperforms all other possible sets of g teams, such set of g teams may be considered an optimal set of g teams. Thus, for each designed data set, there is at least one optimal set of g teams with a fitness level equal to 0.

Data set	Number of topics (k)	Number of participating students (n)	Number of teams (g)
1	6	18	3
2	6	24	4
3	6	60	10
4	6	120	20
5	6	360	60
6	6	600	100
7	6	1200	200
8	6	1800	300
9	6	2400	400

Table 1: Characteristics of data sets.

The steady-state evolutionary algorithm has been run 20 times on each of the nine data sets. As a result of each run, the steady-state evolutionary algorithm provided the best set of g teams of the last population or generation. Then, for each of the nine data sets, the average fitness value of the obtained sets of g teams was calculated, and also the average computation time of the runs was calculated.

The parameter setting used for the above-mentioned experiments is presented in Table 2. This parameter setting was chosen based on preliminary experiments. In this respect, various parameter settings were examined on each data set 10 times and then the parameter setting presented in Table 2 was chosen because this setting reached the best and most stable results.

Table 3 presents the results obtained by the steady-state evolutionary algorithm for each of the nine data sets. The first column presents the name of each data set; the second column presents the average fitness value of the obtained sets of g teams for each data set; and the third column presents the average computation time of the runs performed on each data set. The experiments were performed on a personal computer Intel Core 2 Duo at 3.00 GHz and 4 GB RAM under Windows XP Professional Version 2002. The algorithm has been implemented in Java programming language.

Parameter	Value
Population Size	100
Number of generations	700
Parent selection process	
t (tournament size)	2
Crossover process	
Crossover Probability P_c	0.9
Mutation process	
Mutation Probability P_m	0.2
Survival selection process	
λ (replacement factor)	80

Table 2: Parameter setting used for the steady-state evolutionary algorithm.

Data set	Fitness value	Time (seconds)
1	0	0.6537
2	0	1.4741
3	0	5.922
4	0	15.802
5	0	41.8722
6	0.0402	56.7548
7	0.0601	186.9964
8	0.0902	345.453
9	0.12	516.969

Table 3: Results obtained by the steady-state evolutionary algorithm for each data set

The results in Table 3 are analyzed below considering that, as was previously mentioned, each of the nine data sets has at least one optimal set of g teams with a fitness level equal to 0. This fitness level is considered here as a reference level to evaluate the effectiveness of the steady-state evolutionary algorithm on each data set.

In relation to the average fitness value obtained by the algorithm for each data set, we may mention the following points. For each of the first five data sets, the algorithm has achieved an average fitness value that is equal to 0. This means that, for each of the first five data sets, the algorithm has achieved an optimal set of *g* teams in each of the runs. For each of the last four data sets, the algorithm has achieved an average fitness value that is lower than or equal to 0.12. This means that, for the last four data sets, the algorithm has achieved near-optimal sets of *g* teams. We analyzed the composition of the obtained sets of *g* teams for the last four data sets. Based on this analysis, we may say that each one of these sets of *g* teams contains a very high percentage of teams with no difference in understanding level and interest level with respect of each topic. Based on the results above-mentioned, it is considered that the algorithm has reached very high-quality sets of *g* teams for each of the nine data sets. This suggests that the algorithm may be considered in educational environments to build well-balanced teams regarding the understanding levels and interest levels of their student members in respect of each topic.

In relation to the average computation time required by the algorithm for each data set, we may mention the following points. For the first five data sets, the average time required by the algorithm was lower than 42 seconds. For the last four data sets, the average time required by the algorithm was higher than 56 seconds and lower than 517 seconds. The relevance of the time required by the algorithm depends on the response time required by the educational environment. In some educational environments (e.g., face-to-face courses), the response time is not critical [Alberola et al., 2016]. Thus, the time required by the algorithm for the data sets corresponding to such environments may be considered as acceptable for these environments. In some other educational environments (e.g., on-line and on-demand courses with more than 1000 students), the response time becomes highly relevant [Moreno et al, 2012]. Thus, it would be convenient to decrease the time required by the algorithm for the data sets corresponding to such environments, so that the algorithm can give a more acceptable response time for these environments. In this respect, the time required by the algorithm depends on the hardware and software configuration in which the algorithm is run. The computational experiments presented were developed using an average desktop computer; however, with a higher-performance computer, the time required by the algorithm would be reduced, and the good fitness levels obtained by the algorithm would be preserved. Besides, further research should be conducted in order to decrease the time required by the algorithm, preserving or improving the high fitness levels reached.

5.1 Comparison with a Competing Algorithm

In this section, we compare the performance of the steady-state evolutionary algorithm with that of the particle swarm optimization algorithm presented in [Lin et al., 2010] for solving the addressed problem. To the best of our knowledge, the algorithm presented in [Lin et al., 2010] is the only algorithm previously proposed in the literature for solving the addressed problem.

For sake of simplicity, the algorithm presented in [Lin et al., 2010] will be referred as algorithm SPOA. Like the steady-state evolutionary algorithm, the algorithm SPOA is a population-based stochastic search and optimization algorithm. However, in contrast with the steady-state evolutionary algorithm, the algorithm SPOA is not an evolutionary algorithm. In this respect, the framework of the algorithm SPOA corresponds to a classical particle swarm optimization framework.

Thus, the algorithm SPOA does not include evolutionary operators such as parent selection, crossover, mutation, and survival selection.

In order to compare the performance of the steady-state evolutionary algorithm with that of the algorithm SPOA, the performance of the algorithm SPOA was evaluated on the nine data sets presented in Table 1. Specifically, the algorithm SPOA was run 20 times on each of the nine data sets presented in Table 1, considering the parameter setting recommended in [Lin et al., 2010] for this algorithm. Then, for each of the nine data sets, the average fitness value of the obtained sets of *g* teams and the average computation time of the runs were calculated.

Table 4 presents the results obtained by the algorithm SPOA for each of the nine data sets. Besides, this table contains the results obtained by the steady-state evolutionary algorithm for each of the nine data sets, as were presented in Table 3. The experiments corresponding to both algorithms were performed on a personal computer Intel Core 2 Duo at 3.00 GHz and 4 GB RAM under Windows XP Professional Version 2002.

Data set	Algorithm SPOA		Steady-state evolutionary algorithm	
	Fitness value	Time (seconds)	Fitness value	Time (seconds)
1	0.019	0.1722	0	0.6537
2	0.026	0.3738	0	1.4741
3	0.0459	3.0156	0	5.922
4	0.0391	5.3676	0	15.802
5	0.0476	9.07725	0	41.8722
6	0.081	13.419	0.0402	56.7548
7	0.108	44.15565	0.0601	186.9964
8	0.1513	80.28405	0.0902	345.453
9	0.1955	117.18105	0.12	516.969

Table 4: Results obtained by the algorithm SPOA and results obtained by the steady-state evolutionary algorithm.

In relation to the results presented in Table 4, we may mention the following points. The average fitness value obtained by the steady-state evolutionary algorithm for each data set is significantly better than that obtained by the algorithm SPOA. In particular, the steady-state evolutionary algorithm has obtained optimal fitness values (i.e., optimal sets of *g* teams) in all runs developed on the first five data sets, whereas the algorithm SPOA has not obtained optimal fitness values for these data sets (i.e., the algorithm SPOA has obtained average fitness values over the range [0.019, 0.05] for the first five data sets). These results indicate that the quality of the sets of *g* teams achieved by the steady-state evolutionary algorithm for the data sets is significantly better than that of the sets of *g* teams achieved by the algorithm SPOA. In relation to the time required by the algorithms, the average time required by the steady-state evolutionary algorithm for each data set is higher than that required by the algorithm SPOA. In this respect, as was previously mentioned, the time required by the steady-state evolutionary algorithm is acceptable for some educational environments (e.g., face-to-face courses); however, it would be convenient to reduce the time required by the steady-state evolutionary algorithm for some other educational environments (e.g., online courses with more than 1000 students).

From the analysis of the results presented in Table 4, it may be stated that the steady-state evolutionary algorithm achieved higher-quality sets of g teams than the algorithm SPOA for each data set. This is mainly because of, in contrast with the algorithm SPOA, the steady-state evolutionary algorithm uses different evolutionary processes to search the solution spaces inherent to the problem instances represented by the data sets. Specifically, the steady-state evolutionary algorithm uses a parent selection process, a crossover process, a mutation process and a survival selection process. The selection processes guide the evolutionary search towards the higher-quality regions of the solution space, selecting the sets of g teams that belong to the higher-quality regions. The crossover process explores the regions indicated by the selection processes, generating sets of g teams from pairs of sets of g teams selected by the selection processes. The mutation process fine-tunes the search developed by the crossover process on the regions indicated by the selection processes, fine-tuning the sets of g teams obtained by the crossover process. Because of the use of these evolutionary processes, the steady-state evolutionary algorithm can reach higher-quality sets of g teams than the algorithm SPOA for the nine data sets.

Based on the above-mentioned, the steady-state evolutionary algorithm may be considered in educational environments to build sets of teams much better balanced than algorithm SPOA, regarding the understanding levels and interest levels of their student members in respect of each topic.

6 Conclusions and Future Work

In this paper, we have addressed the collaborative learning team building problem described in [Lin et al., 2010]. This problem considers two grouping criteria that must be simultaneously satisfied. One of these criteria considers the understanding levels of the students in respect of the topics of the course, and is based on building well-balanced teams regarding the understanding levels of their members. The other criterion considers the interest levels of the students in respect of the topics of the course, and is based on building well-balanced teams regarding the interest levels of their members. These two grouping criteria are usually considered by teachers in real-world classrooms [Saleh and Kim, 2009]. Besides, these two grouping criteria have been successfully evaluated in many different kinds of educational environments [Michaelsen et al., 2004; Yang, 2006; Saleh and Kim, 2009; Nielsen et al., 2009]. Thus, it is considered that the collaborative learning team building problem described in [Lin et al., 2010] is really valuable in the context of collaborative learning.

To solve the addressed problem, we have proposed a steady-state evolutionary algorithm. This algorithm explores different sets of teams which may be defined from the students taking the course, with the aim of finding the sets of teams that optimize the two grouping criteria. The explored sets of teams are evaluated with respect of the two grouping criteria. In order to perform that evaluation, the algorithm is based on knowledge of the understanding levels and interest levels of the students with respect of each of the topics of the course.

We have presented the computational experiments carried out to evaluate the performance of the proposed steady-state evolutionary algorithm. In this respect, the performance of the steady-state evolutionary algorithm was evaluated on nine data sets corresponding to different instances of the addressed problem. These instances have different levels of complexity. Then, the performance of the steady-state evolutionary algorithm was compared with that of the particle swarm optimization algorithm proposed in [Lin et al., 2010] for solving the addressed problem. To the best of our knowledge, the algorithm presented in [Lin et al., 2010] is the only algorithm previously proposed in the literature for solving the addressed problem.

Based on the results obtained by the steady-state evolutionary algorithm for the nine data sets, we may state that this algorithm has reached very high-quality sets of teams for each of the data sets. Specifically, the algorithm has reached optimal sets of teams (i.e., sets of teams with a fitness equal to 0) for the five less complex data sets, and near-optimal sets of teams (i.e., sets of teams with an average fitness lower than or equal to 0.12) for the four more complex data sets. In relation to the time required by the algorithm, we consider that the time required by the algorithm is acceptable for some educational environments (e.g., face-to-face courses), however it would be convenient to decrease the time required by the algorithm for some other educational environments (e.g., online courses with more than 1000 students). Besides, as a result of the comparative analysis conducted, we may state that the steady-state evolutionary algorithm achieved much higher-quality sets of teams than the particle swarm optimization algorithm proposed in [Lin et al., 2010] for each of the nine data sets. Thus, the steady-state evolutionary algorithm may be considered in educational environments to build sets of teams much better balanced than particle swarm optimization algorithm proposed in [Lin et al., 2010], regarding the understanding levels and interest levels of their student members in respect of each topic.

In future works, we will incorporate other relevant grouping criteria into the addressed problem and then we will adapt the fitness function of the proposed steady-state evolutionary algorithm according to these grouping criteria. On the other hand, in future works, we will investigate the hybridization of the steady-state evolutionary algorithm with other search and optimization algorithms (e.g., simulated annealing algorithms, hill-climbing algorithms, and tabu search algorithms), in order to decrease the time required by the algorithm, and preserve or improve the high fitness levels reached by the algorithm. Moreover, we will evaluate other crossover and mutation processes for the encoding of solutions used in the steady-state evolutionary algorithm. In addition, we will evaluate other survival selection processes proposed in the literature (e.g., deterministic crowding).

References

[Alberola et al., 2016] Alberola, J., Del Val, E., Sanchez-Anguix, V., Palomares, A., Teruel, M: "An artificial intelligence tool for heterogeneous team formation in the classroom"; Knowledge-Based Systems, 101, 1 (2016), 1-14.

[Barkley et al., 2005] Barkley, E.F., Cross, K.P., Howell Major, C.: "Collaborative learning techniques"; John Wiley & Sons, Inc. (2005)

[Christodoulopoulos and Papanikolaou, 2007] Christodoulopoulos, C.E., Papanikolaou, K.A.: "A Group Formation Tool in an E-Learning Context"; Proc. 19th IEEE ICTAI 2007, IEEE Press, New York (2007), 117–123.

[Cruz and Isotani, 2014] Cruz, W. M., Isotani, S.: "Group Formation Algorithms in Collaborative Learning Contexts: A Systematic Mapping of the Literature"; Lecture Notes in Computer Science, 8658 (2014), 199-214.

[Deb, 2009] Deb, K.: "Multi-objective optimization using evolutionary algorithms"; Wiley, NewYork (2009)

[Eiben and Smith, 2015] Eiben, A. E., Smith, J. E.: "Introduction to evolutionary computing" (2nd ed.); Springer, Germany (2015)

[Graf and Bekele, 2006] Graf, S., Bekele, R.: "Forming Heterogeneous Groups for Intelligent Collaborative Learning Systems with Ant Colony Optimization"; Lecture Notes in Computer Science, 4053 (2006), 217–226.

[Isotani et al., 2009] Isotani, S., Inaba, A., Ikeda, M., Mizoguchi, R.: "An ontology engineering approach to the realization of theory-driven group formation"; International Journal of Computer-Supported Collaborative Learning, 4, 4 (2009), 445–478

[Kachitvichyanukul, 2012] Kachitvichyanukul, V.: "Comparison of Three Evolutionary Algorithms: GA, PSO, and DE"; Industrial Engineering & Management Systems, 11, 3 (2012), 215-223.

[Lin et al., 2010] Lin, Y. T., Huang, Y. M., Cheng, S. C.: "An automatic group composition system for composing collaborative learning groups using enhanced particle swarm optimization"; Computers and Education, 55 (2010), 1483–1493.

[Meyer, 2009] Meyer, D.: "OptAssign - A web-based tool for assigning students to groups"; Computers and Education, 53 (2009), 1104–1119.

[Michaelsen et al., 2004] Michaelsen, L. K., Knight, A. B., Fink, L. D.: "Team-based learning: A transformative use of small groups in college teaching"; Stylus Publishing, Sterling, VA (2004)

[Moreno et al., 2012] Moreno, J., Ovalle, D., Vicari, R.: "A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics"; Computers & Education, 58, 1 (2012), 560–569

[Nielsen et al., 2009] Nielsen, T., Hvas, A. E., Kjaergaard, A.: "Student team formation based on learning styles at university start: does it make a difference to the student ?"; Reflection Education, 5, 2 (2009), 85–103.

[Ounnas, 2010] Ounnas, A.: "Enhancing the Automation of Forming Groups for Education with Semantics"; University of Southampton, Southampton (2010)

[Saishanmuga Raja and Rajagopalan, 2014] Saishanmuga Raja, V., Rajagopalan, S. P.: "A comparative analysis of optimization techniques for artificial neural network in bio medical applications"; Journal of Computer Science, 10, 1 (2014), 106-114.

[Saleh and Kim, 2009] Saleh, I., Kim, S.: "A fuzzy system for evaluating students' learning achievement"; Expert Systems with Applications, 36, 3 (2009), 3243–6236.

[Wang et al., 2007] Wang, D.Y., Lin, S.S.J., Sun, C.T.: "DIANA: A computer-supported heterogeneous grouping system for teachers to conduct successful small learning groups"; Computers in Human Behaviors, 23, 4 (2007), 1997–2010.

[Yang, 2006] Yang, S. J. H.: "Context aware ubiquitous learning environments for peer-to-peer collaborative learning"; Journal of Educational Technology & Society, 9, 1 (2006), 188–201.

[Yannibelli and Amandi, 2012a] Yannibelli, V., Amandi, A.: “A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context”; Expert Systems with Applications, 39, 10 (2012), 8584–8592.

[Yannibelli and Amandi, 2012b] Yannibelli, V., Amandi, A.: “A memetic algorithm for collaborative learning team formation in the context of software engineering courses”; Lecture Notes in Computer Science, 7547 (2012), 92–103.

[Yannibelli and Amandi, 2013] Yannibelli, V., Amandi, A.: “A hybrid algorithm combining an evolutionary algorithm and a simulated annealing algorithm to solve a collaborative learning team building problem”; Lecture Notes in Computer Science, 8073 (2013), 376–389.

[Zhamri Che Ani et al., 2010] Zhamri Che Ani, Azman Yasin, Mohd Zabidin Husin, Zauridah Abdul Hamid: “A method for group formation using genetic algorithm”; International Journal on Computer Science and Engineering, 2, 9 (2010), 3060–3064.