Expert Systems with Applications 39 (2012) 8584–8592

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context

Virginia Yannibelli *, Analía Amandi

ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Tandil, Buenos Aires, Argentina
CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina
Campus Universitario, Paraje Arroyo Seco, Tandil 7000, Buenos Aires, Argentina

ARTICLE INFO

Keywords:
Collaborative learning
Learning team formation
Roles
Evolutionary algorithms

ABSTRACT

The aim of forming collaborative learning teams is that participating students acquire new knowledge and skills through the interaction with their peers. To reach this aim, teachers usually utilize a grouping criterion based on the students' roles and on forming well-balanced teams according to the roles of their members. However, the implementation of this criterion requires a considerable amount of time, effort and knowledge on the part of the teachers. In this paper, we propose a deterministic crowding evolutionary algorithm with the aim of assisting teachers when forming well-balanced collaborative learning teams. Considering a given number of students who must be divided into a given number of teams, the algorithm both designs different alternatives to divide students into teams and evaluates each alternative as regards the grouping criterion previously mentioned. This evaluation is carried out on the basis of knowledge of the students' roles. To analyze the performance of the proposed algorithm, we present the computational experiments developed on ten data sets with different levels of complexity. The obtained results are really promising since the algorithm has reached optimal solutions for the first four data sets and near-optimal solutions for the remaining six data sets.

1. Introduction

In educational centers, teachers usually divide students into learning teams to perform collaborative learning tasks. These tasks require students to work together to solve problems, discover information, and complete projects. Besides, the collaborative learning tasks are meant for students to acquire new knowledge and skills through the interaction with their peers. Thus, the objective of collaborative learning is to supplement and enrich individual learning (Barkley, Cross, & Howell Major, 2005; Michaelsen, Knight, & Fink, 2004).

Within a collaborative learning environment, there are two important aspects to take into account: the criterion to form learning teams (i.e., grouping criterion) and the way in which the grouping criterion is applied (i.e., either manually or automatically). The grouping criterion is relevant since the way in which a team is made up affects both the learning level and the social behavior of the students belonging to the team (Beane & Lemke, 1971; Dalton, Hannafin, & Hooper, 1989; Hooper & Hannafin, 1988; Michaelsen et al., 2004; Webb, 1982).

Several grouping criteria have been proposed so far in order to form collaborative learning teams. Generally, these criteria take into account factors related to the learning state of the students, their learning style, personality, and interpersonal relationships (Alfonseca, Carro, Martín, Ortigosa, & Paredes, 2006; Lin, Huang, & Cheng, 2010; Martin & Paredes, 2004; Meyer, 2009; Michaelsen et al., 2004; Nielsen, Hvas, & Kjaergaard, 2009; Rutherfoord, 2001; Saleh & Kim, 2009; Sánchez Hórreo & Carro, 2007; Speck, 2003; Tang, Chan, Winoto, & Wu, 2001; Wilkinsona & Fung, 2002; Yang, 2006).

One of the grouping criteria most utilized by teachers in classrooms is based on taking into account the students' roles and on forming well-balanced teams according to the roles of their members. A role is the way in which a person tends to behave, contribute and interrelate with others throughout a collaborative task. Several team role models proposed in the literature recommend this grouping criterion (Belbin, 1981, 1993; Davis, Millburn, Murphy, & Woodhouse, 1992; Margerison & McCann, 1990; Parker, 1990; Spencer & Pruss, 1992; Woodcock, 1989). According to these models, if a team is well-balanced with respect to the roles of its members, then its members will interact and collaborate with each other well. These models also indicate that there is a direct relationship between the performance of a team and the balance level

* Corresponding author at: ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Campus Universitario, Paraje Arroyo Seco, Tandil (7000), Buenos Aires, Argentina. Tel.: +54 (02293) 439682x28; fax: +54 (02293) 439681x52.

E-mail addresses: vyannibe@exa.unicen.edu.ar (V. Yannibelli), amandi@exa.unicen.edu.ar (A. Amandi).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2012.01.195

among the roles of its members. Among the models developed so far, the model proposed by Belbin (Belbin, 1981, 1993) is the most widespread and accepted in the literature.

The Belbin's model (Belbin, 1981, 1993) has been successfully used in education as a tool for collaborative learning team formation (Jeffries, Grodzinsky, & Griffin, 2003, 2004; Johansen, 2003; McFadzean, 2001; Ounnas, 2010; Stevens, 1998; Winter, 2004). Different works that employed the Belbin's model to study teams of students tasking software engineering group projects showed that considering the Belbin's roles can impact positively on the performance of the teams (Stevens, 1998; Winter, 2004), and can provide a prediction of the performance of the teams based on the composition of the roles within the teams (Johansen, 2003). Besides, the Belbin's model (Belbin, 1981, 1993) has been widely used in training activities by many organizations, consulting firms and executive education programs (Cameron, 2002; Park & Bang, 2002; Prichard & Stanton, 1999; Sommerville & Dalziel, 1998).

The formation of well-balanced collaborative learning teams according to the roles of their members is a task requiring a considerable amount of time, effort and knowledge of the roles of the students. This task is not free from mistakes made by teachers due to their human limitations. Some examples of such limitations are lack of knowledge and wrong assumptions regarding the roles of the students and the formation of well-balanced teams. Therefore, it is valuable to assist teachers when forming collaborative learning teams. In this line of thought, the aim of automatically forming learning teams is to build teams efficiently (i.e., the time required by automation is shorter than the time required by teachers) and effectively (i.e., wrong decisions are minimized when all the available knowledge of students' roles and the formation of well-balanced teams is considered). Through automation it is thus possible to considerably reduce the work load of teachers and optimize the formation of collaborative learning teams.

In this paper, we address the problem of forming collaborative learning teams automatically. As part of the problem, we consider that teams must be made up in such a way that the balance among the roles of their members is maximized. This grouping criterion is defined on the basis of the model proposed by Belbin (Belbin, 1981, 1993) and the balance conditions established by this author. The reason for this is that different works have showed that the collaborative learning team formation based on the model proposed by Belbin impact positively on the learning level and the social behavior of the students belonging to the teams and on the performance of the teams (Jeffries et al., 2003; Jeffries, Grodzinsky, & Griffin, 2004; Johansen, 2003; McFadzean, 2001; Ounnas, 2010; Stevens, 1998; Winter, 2004).

In order to solve the problem, we propose a deterministic crowding evolutionary algorithm. Taking into account a given number of students who must be divided into a given number of teams, the algorithm both designs different alternatives to divide students into learning teams and evaluates each alternative as regards the grouping criterion considered as part of the problem. That evaluation is carried out on the basis of knowledge of the students' roles.

We have decided to propose an evolutionary algorithm because the problem addressed in this paper is an *NP-Hard* optimization problem, and in this sense, evolutionary algorithms have been proved to be effective and efficient when resolving a wide variety of *NP-Hard* optimization problems (Eiben & Smith, 2007; Goldberg, 2007).

The rest of the paper is organized as follows. Section 2 presents a description of the problem addressed. Section 3 describes the evolutionary algorithm designed to solve the problem. Section 4 presents the computational experiments carried out to evaluate the evolutionary algorithm and an analysis of the results obtained. Finally, Section 5 presents the conclusions of this work.

2. Problem description

In this section, the problem addressed is formally described. This problem consists in forming teams of students to perform collaborative learning tasks. As part of the problem, we consider that teams must be made up in such a way that the balance among the roles of their members is maximized.

A class S is made up of n students, $S = \{s_1, s_2, \ldots, s_n\}$. The teacher must divide the n students into g teams, $G = \{G_1, G_2, \ldots, G_g\}$. Each G_i team is made up of a z_i number of member students, and each student can only belong to one team. As regards team size, students must be divided in such a way that the g 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 S, n, and g are known.

As regards the students, they naturally assume or play different roles when taking part in a collaborative task. A role is the way in which a person tends to behave, contribute and interrelate with others throughout a collaborative task. In relation to the roles that can be played by the students, the nine roles defined in Belbin's model (Belbin, 1981, 1993) are taken into account in this work. Table 1 shows the nine roles and a brief description of the features of each. Belbin (Belbin, 1981, 1993) considers that each person has a preference level for each role, and in this sense, defines four preference levels: low, average, high and very high. The preference level indicates how naturally a person can play a given role. Then, Belbin (Belbin, 1981, 1993) considers that if a person has a high or very high preference level for a given role, that person is capable of playing that role naturally. Furthermore, Belbin (Belbin, 1981, 1993) points out that a person can play one or several roles naturally.

According to Belbin's model (Belbin, 1981, 1993), each student naturally plays one or several of the nine roles described in Table 1. In this respect, the roles naturally played by each student are known data. These roles are obtained through the Belbin Team-Role Self-Perception Inventory (BTRSPI) developed by Belbin (Belbin, 1981, 1993). The BTRSPI determines the preferred team roles of the persons by giving them self-evaluation tests (Belbin, 1981, 1993).

As previously mentioned, teams must be made up in such a way that the balance among the roles of their members is maximized. This grouping criterion requires analyzing the balance level in the formed teams. In order to analyze the balance level, the balance conditions established by Belbin are considered (Belbin, 1981, 1993). These conditions are presented below.

Belbin (Belbin, 1981, 1993) states that a team is balanced if each role specified in his model is played naturally by at least one team member. In other words, in a balanced team, all team roles are naturally played. Further, Belbin states that each role should be naturally played by only one team member (Belbin, 1981). Belbin states that a team is unbalanced if some roles are not played naturally or if several of its members play the same role naturally (i.e., duplicate role) (Belbin, 1981, 1993).

Formulas (1) and (2) have been designed in order to formally express the balance conditions established by Belbin (Belbin, 1981, 1993). Formula (1) analyzes the way in which a given r role is played within a given G_i team and gives a score accordingly. If r is naturally played by only one member of G_i team, then 1 point is awarded to G_i. Conversely, if r is not naturally played by any member of G_i, or otherwise r is naturally played by several members of G_i, then 2 points and p points are taken off respectively.

Formula (2) sets the balance level in a given G_i team. This balance level is established based on the scores obtained by G_i, through Formula (1), in relation to the nine roles. In this way, the greater the number of non-duplicate roles (i.e., roles played

Table 1
Belbin's role characteristics.

Role	Characteristics
Implementer (IM)	Concerned with the practical translation and application of concepts and plans developed by the team. This entails a down-to-earth outlook, coupled with perseverance in the face of difficulties
Co-ordinator (CO)	Organises, co-ordinates and controls the activities of the team. This involves the clarification of team objectives and problems, assigning tasks and responsibilities, and encouraging team members to get involved in achieving objectives and goals
Shaper (SH)	Challenges, argues and disagrees. Is achievement-motivated, extrovert, impatient, and has a low frustration threshold. Keen on winning the game. Has good insight, especially if loses. A non-chair leader
Plant (PL)	Concerned with putting forward ideas and strategies for achieving the objectives adopted by the team. Performance of this role requires creativity, imagination and innovation
Resource Investigator (RI)	Explores the environment outside the team, by identifying ideas, information and resources. Performance of this role involves developing contacts, co-ordination and negotiation with other teams and individuals
Monitor Evaluator (ME)	Analyses ideas and proposals being considered by the team, to evaluate their feasibility and value for achieving the team's objectives. Points out in a constructive manner the weaknesses of proposals being considered
Team Worker (TW)	Creates and maintains a team spirit. This involves improving communication by providing personal support and warmth to team members and by overcoming tension and conflict
Completer/Finisher (CF)	Ensures that the team's efforts achieve appropriate standards, and that mistakes of both commissions and omissions are avoided. It also involves searching for detailed mistakes and maintaining a sense of urgency within the team
Specialist (SP)	Provides knowledge and skills in key areas. Single-minded, self-starting, dedicated

naturally by only one member of G_i), the greater the balance level assigned to G_i. Conversely, the fewer the number of roles played naturally, or the more duplicate roles, the lower the balance level assigned to G_i. The balance conditions established by Belbin (Belbin, 1981, 1993) can be seen in Formula (2). Using this formula, a perfectly balanced team will obtain a level equal to 9.

$$nr(G_i, r) = \begin{cases} 1 & \text{if } r \text{ is naturally played by only one member of } G_i \\ -2 & \text{if } r \text{ is not naturally played in } G_i \\ -p & \text{if } r \text{ is naturally played by } p \text{ members of } G_i \end{cases} \quad (1)$$

$$nb(G_i) = \sum_{r=1}^{9} nr(G_i, r) \quad (2)$$

Formula (3) has been designed in order to formally express the grouping criterion considered as part of the problem. This formula maximizes the average balance level of g teams defined from the n students in the class. In other words, the objective of this formula is to find a solution (i.e., set of g teams) that maximizes the average balance level of g teams. This is the optimal solution to the problem addressed. In Formula (3), set C contains all the sets of g teams that may be defined from the n students in the class. The term G represents a set of g teams belonging to C. The term $b(G)$ represents the average balance level of the g teams belonging to set G. Then, Formula (3) uses Formula (2) to establish the balance level of each G_i team belonging to the G set.

$$\max_{\forall G \in C} \left(b(G) = \frac{\sum_{i=1}^{g} nb(G_i)}{g} \right) \quad (3)$$

3. Deterministic crowding evolutionary algorithm

To solve the problem addressed in this paper, we propose a deterministic crowding evolutionary algorithm. Evolutionary algorithms are heuristic methods of search and optimization inspired by Darwin's theory of evolution (Eiben & Smith, 2007; Goldberg, 2007). According to these algorithms, an initial population of candidate solutions to a problem evolves towards the optimal solutions based on the principles of natural selection, crossover, and mutation.

The general behavior of the algorithm proposed here is as follows. Considering a class of n students who shall be divided into g teams, the algorithm starts the evolution from an initial population of feasible solutions. Each of these solutions codifies a feasible set of g teams which may be defined when the n students are divided. Then, each solution of the population is decoded (i.e., the set of g teams inherent to the solution is built) and evaluated by a fitness function. This function evaluates each solution in relation to the optimization objective of the problem. As explained earlier, the objective here is to maximize the balance level of the g teams formed from n students. Therefore, taking into consideration a given solution, the function evaluates the balance level of the g teams represented by the solution. In order to perform that evaluation, the function is based on knowledge of the students' roles.

Once the solutions are evaluated, a selection process is applied to the current population. Some solutions of the population are selected and then paired. In general, the solutions with the greatest fitness values have more chances of being selected. Then a crossover process is applied to each pair of solutions to generate new feasible ones. A mutation process is later applied to the generated solutions by the crossover. This mutation process is aimed at introducing genetic diversity in solutions. Finally, a strategy known as deterministic crowding (Eiben & Smith, 2007; Goldberg, 2007) is used to create a new population from the solutions in the current population and the new generated solutions.

This process is repeated until some stopping criterion is reached. In this case, the stopping criterion consists in reaching a predetermined number of repetitions or iterations.

Details about each of the different components of the proposed algorithm are presented in the next subsections. The main components of the algorithm are the representation of solutions, the generation of the initial population, the fitness function, and the selection, crossover, and mutation processes.

3.1. Representation of the solutions

Each solution in the evolutionary algorithm population represents or encodes a G set of g teams which may be built when the n students in the class are divided. The solutions must be encoded in such a way that the application of different crossover and mutation operators generates new feasible solutions. Therefore, we define an appropriate encoding for the solutions below.

Each solution is encoded as a list with a length equal to n (i.e., a list with as many positions as students in the class). Specifically, each position j ($j = 1, \ldots, n$) on this list contains a different student (i.e., repeated students are not admitted). Besides, each student s_k ($k = 1, \ldots, n$) may be in any position on the list. In short, the list is a permutation of the n students.

3.1.1. Decoding of the Solutions

We propose here a decoding process through which g teams of students may be built from the above-mentioned representation (i.e., list of the n students or permutation of the n students). In this process, the g teams are built taking into account the two restrictions considered as part of the problem. The first restriction is that each student may belong to only one team. While the second restriction holds 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 is as follows.

Firstly, in this process the size of the teams is considered to depend on the relationship between the values n and g. Therefore, the process starts by calculating the value of the term $z = (n/g)$. If z is an integer, then g teams having the same size are built. Each team G_i ($i = 1,\ldots,g$) is assigned a size $z_i = z$. Then, the process defines how each team is made up. The composition of a G_i team is defined based on its z_i size, its i index and the above-mentioned list of the n students. Specifically, a set of z students is assigned to each G_i team. The set assigned to each G_i is made up of the students in positions $[((i-1)\times z+1),\ldots,(i\times z)]$ on the list.

In case z is not an integer, g teams having the same size cannot possibly be built. Furthermore, the process must consider 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. The way in which the g teams are built is described below.

Firstly, considering that z is not an integer (i.e., z is a real number) and that the above-mentioned restriction should be taken into account, some of the g teams will have a size $z_1 =$ ((integer part of z) + 1) and the rest of the teams will have a size $z_2 =$ (integer part of z). Then the process defines which teams will have size z_1 and which teams will have size z_2. Specifically, the process assigns each team G_l ($l = 1,\ldots,g_1$) a size $z_l = z_1$ considering $g_1 = (n - ((\text{integer part of } z)\times g))$. Then the process assigns each team G_t ($t = (g_1+1),\ldots,g$) a size $z_t = z_2$. Lastly, the composition of the g teams is determined based on points (a), (b) and (c) below.

(a) The process assigns each team G_l ($l = 1,\ldots,g_1$) a set of z_1 students. The set of students assigned to each G_l is made up of the students in positions $[((l-1)\times z_1+1),\ldots,(l\times z_1)]$ on the list of the n students.
(b) The process assigns team G_r ($r = g_1+1$) a set of z_2 students. The set assigned to G_r is made up of the students in positions $[(g_1\times z_1)+1,\ldots,(g_1\times z_1)+z_2]$ on the list of the n students.
(c) The process assigns each team G_f ($f = g_1+2,\ldots,g$) a set of z_2 students. The set assigned to each G_f is made up of the students in positions $[((g_1\times z_1)+z_2)+((f-(g_1+2))\times z_2+1),\ldots,((g_1\times z_1)+z_2)+((f-(g_1+1))\times z_2)]$ on the list.

Fig. 1 shows a feasible encoded solution to an example of the problem addressed in this paper. In this example, 7 students have to be divided into 3 teams. The encoded solution is a list of the 7 students or a permutation of the 7 students. This figure also shows the 3 teams obtained through the application of the decoding process.

3.2. Initial population

The initial population contains a specific number of feasible encoded solutions to the problem. Each encoded solution consists of a permutation of the n students in the class (Subsection 3.1). A random method has been designed so as to generate each of the solutions of this population. This kind of methods guarantees a good level of genetic diversity in the initial population, and therefore, helps prevent the premature convergence of the algorithm (Eiben & Smith, 2007; Goldberg, 2007).

The method begins as follows: an empty list of length n is built. Then, n iterations are developed so as to define the content of the positions on the list. In each iteration m ($m = 1,\ldots,n$), a student from the L_m set is randomly selected and positioned at m on the list. The L_m set is made up of those students in the class who, up to the m iteration, have not been included on the list. A permutation of the n students in the class is therefore generated.

3.3. Fitness function

It evaluates the fitness level of a given solution in relation to the optimization objective considered as part of the problem. As explained earlier, the objective here is to maximize the balance level of the g teams formed from the n students in the class.

Considering a given encoded solution, the function decodes the G set of g teams represented by the solution. The decoding is carried out by applying the process described in Subsection 3.1.1. Then, the function calculates the value of the term $b\,(G)$ (Formulas (3), (2) and (1)). This value represents the average balance level of the g teams composing the G set, and thus, determines the fitness level of the encoded solution. In the case of a G set of perfectly balanced g teams, the value of the term $b\,(G)$ shall be 9, considering 9 as the maximal possible fitness level.

In order to calculate the value of the term $b\,(G)$, the function needs to know the roles inherent to each of the n students in the class. In this sense, each class of n students has an associated knowledge base, and this contains the roles inherent to each student of the class. Then, the fitness function queries the base to obtain the roles of each student.

As was mentioned in Section 2, the roles of the students are obtained through the Belbin Team-Role Self-Perception Inventory (BTRSPI) developed by Belbin (Belbin, 1981, 1993). The BTRSPI determines the preferred team roles of the persons by giving them self-evaluation tests (Belbin, 1981, 1993).

3.4. Selection of parents

This process selects the best solutions from among the current population to conform pairs of solutions (parent solutions), which will be used to generate new solutions (offspring solutions) by the crossover and mutation processes.

When selecting a solution, this process considers the fitness values of the solutions in the population and is usually biased by a random factor. Thus, the solutions with the highest fitness values will have more chances of being selected.

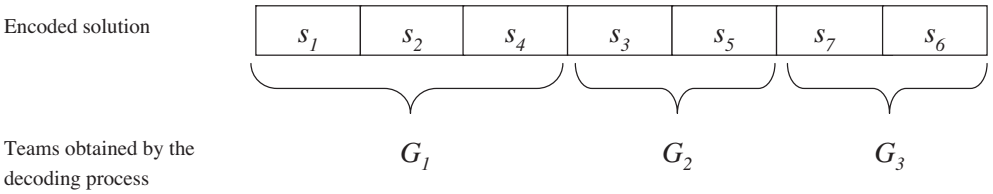

Fig. 1. Feasible encoded solution to an example of the addressed problem, and teams obtained through the application of the decoding process.

In order to develop the parent selection process, we have decided to use a method known as tournament selection (Eiben & Smith, 2007; Goldberg, 2007). This is, in fact, one of the most applied and widespread in the literature. This method is described below.

Two solutions are randomly selected from the current population. Then, the fitness values of those solutions are compared. The better one (i.e., the solution with the best fitness value) is selected and becomes the first member of the pair which is being designed. The other solution is returned to the population. This operation is repeated so as to obtain the second member of the pair. A pair of solutions is thus built out of the current population.

The method described in the previous paragraph is applied $M/2$ times to the current population to obtain $M/2$ pairs, considering M as the size of the population.

3.5. Crossover

The selection process determines a number of pairs of solutions that should be recombined, and each pair undergoes the crossover operator with probability P_c. The crossover developed in a pair of solutions (parent solutions) generates two new solutions (offspring solutions).

The crossover operator is one of the most important genetic operators because it preserves and combines the best characteristics of the parent solutions so that new, better solutions can be defined (Eiben & Smith, 2007; Goldberg, 2007).

The crossover operator is directly applied to pairs of encoded solutions. Thus, the crossover must be designed on the basis of the representation defined for the solutions. In this case, each solution consists of a permutation of n elements and, therefore, it is necessary to use a feasible operator for permutations of n elements. We have decided to use an operator known as order crossover (Eiben & Smith, 2007). This is, in fact, one of the most applied and widespread in the literature. This operator is described below.

The operator is applied to the parent 1 and parent 2 solutions, which generates two new solutions, offspring 1 and offspring 2. The operator defines two random crossover points $k1$ and $k2$, considering $1 \leqslant k1 < k2 < n$, and n is equal to the longitude of the parent solutions. Then, to define offspring 1, the operation is as follows. Firstly, the elements in the positions $[k1, k2]$ of parent 1 are copied, in the same order, in the positions $[k1, k2]$ of offspring 1. Then, the operator copies the elements not included in offspring 1 in the empty positions of this solution. The elements not included in offspring 1 are copied taking into account the order in which they appear in parent 2. Specifically, starting from the second crossover point in parent 2, the operator copies the elements not included in the order in which they appear in parent 2. When the operator reaches the last element on parent 2 list, the process continues from the first position on that list.

The generation of offspring 2 is similar to the generation of offspring 1. However, the roles of the parents are inverted to generate offspring 2.

Fig. 2 shows an example of the crossover operation described. In this example, the operation is applied to two parent solutions with longitude 7, and two offspring solutions are generated. The crossover points $k1$ and $k2$ are equal to 2 and 5 respectively. Fig. 2. (a) shows that the segment $[k1, k2]$ of parent 1 is copied in offspring 1, and the segment $[k1, k2]$ of parent 2 is copied in offspring 2. Fig. 2. (b) shows that the elements not included in offspring 1 are copied in this solution in the order in which they appear in parent 2. Meanwhile, the elements not included in offspring 2 are copied in this solution in the order in which they appear in parent 1.

3.6. Mutation

The mutation operator aims at introducing genetic diversity in the population (Eiben & Smith, 2007; Goldberg, 2007). In order to do so, this operator randomly alters one or more characteristics of some solutions obtained after the crossover process. The mutation operator is applied to each solution obtained by the crossover process with a probability $P_{m.}$

The mutation operator is directly applied to an encoded solution. Therefore, in this case, it is necessary to apply a feasible operator for permutations of n elements. We have decided to use an operator known as swap mutation (Eiben & Smith, 2007; Goldberg, 2007). This operator is described below.

The operator is applied to a solution that must be mutated, thus generating a new solution. This operator starts by randomly selecting two positions $k1$ and $k2$, considering $1 \leqslant k1 < k2 \leqslant n$, and n equal to the longitude of the solution. Then, the operator removes the elements positioned at $k1$ and $k2$ from the solution. Later, the element removed from the position $k1$ is copied in the position $k2$, and the element removed from the position $k2$ is copied in the position $k1$. In short, the mutation operator swaps the elements in positions $k1$ and $k2$. Thus, a new permutation of n elements is generated.

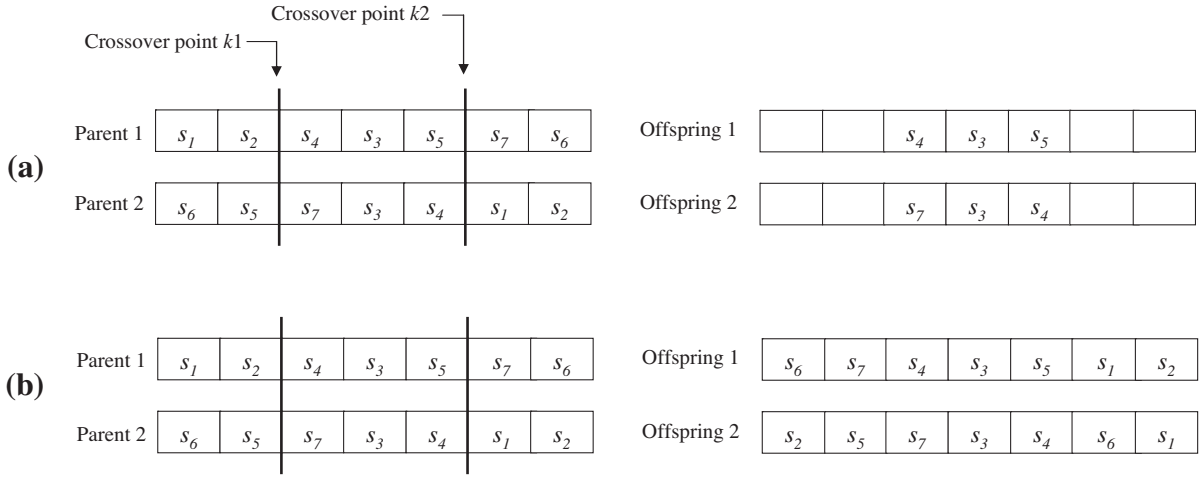

Fig. 2. Example of the crossover operator. **(a)** First step. **(b)** Second step.

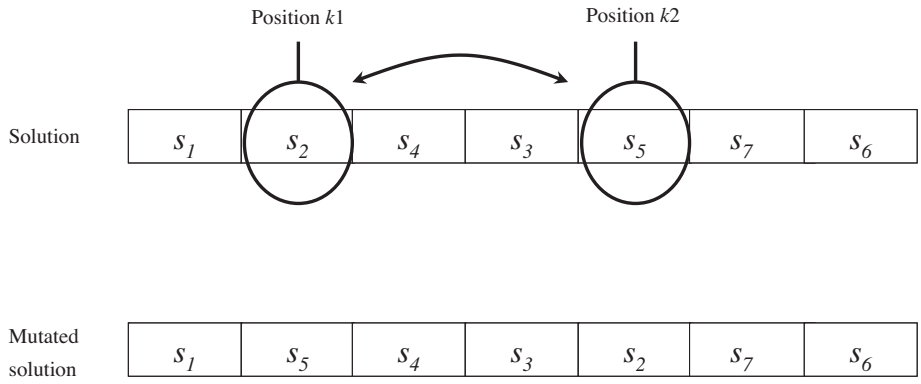

Fig. 3. Example of the mutation operator.

Fig. 3 shows an example of the mutation operation described. In this example, the operation is applied to a solution with longitude 7, and a new solution is generated. The positions $k1$ and $k2$ are equal to 2 and 5 respectively.

4. Computational experiments

This section describes the computational experiments developed to evaluate the effectiveness and efficiency of the evolutionary algorithm. Subsection 4.1 describes the data sets that were utilized. Subsection 4.2 presents the selected values for setting the algorithm parameters. Subsection 4.3 presents and analyzes the results obtained by the algorithm on the different data sets. Subsection 4.4 compares the performance of the evolutionary algorithm with those of two competing algorithms.

4.1. Data sets

In order to develop the experiments, we designed 10 data sets. The main characteristics of each data set are shown in Table 2. Each data set contains a list of n students. For each data set, we established a g number of teams to be built from n students. It should be noted 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).

Besides, the evolutionary algorithm needs to know the roles played by the n students from which the g teams are to be built. Thus, specific roles have been specified for each student of each of the 10 data sets. These roles belong to the Belbin's model (Belbin, 1981, 1993) containing 9 roles (i.e., IM, CO, SH, PL, RI, ME, TW, CF y SP) as described in Table 1. What follows is a description of the procedure employed for assigning roles to the students of each of the 10 data sets shown in Table 2.

Table 2
Description of each data set.

Data set	Number of participating students (n)	Number of teams (g)
1	18	3
2	24	4
3	60	10
4	120	20
5	360	60
6	600	100
7	1200	200
8	1800	300
9	2400	400
10	3000	500

Considering a data set containing a list of $n = g \times 6$ students from which g teams of 6 members are to be built, the procedure for role assignment performs the following 6 steps.

1. The procedure considers students in positions $[1,\ldots,g]$ on the list and assigns each of them role IM.
2. The procedure considers students in positions $[g+1,\ldots,2 \times g]$ on the list and assigns each of them roles CO and SH.
3. The procedure considers students in positions $[(2 \times g)+1,\ldots,3 \times g]$ on the list and assigns each of them roles PL and RI.
4. The procedure considers students in positions $[(3 \times g)+1,\ldots,4 \times g]$ on the list and assigns each of them roles ME and TW.
5. The procedure considers students in positions $[(4 \times g)+1,\ldots,5 \times g]$ on the list and assigns each of them role CF.
6. The procedure considers students in positions $[(5 \times g)+1,\ldots,6 \times g]$ on the list and assigns each of them role SP.

By means of the described procedure, each of the 9 roles is represented by g students of the data set, and each student of the data set represents one or two roles. In this way, from the n students of the data set, it is possible to build at least a set of perfectly balanced g teams. In other words, it is possible to build at least a set of g teams in which each of the nine roles is represented by only one team member. According to the fitness function defined in Subsection 3.3, a set of perfectly balanced g teams has a fitness level (i.e., the average balance level of the g teams) that is equal to 9. This fitness level is the maximal possible fitness level.

Taking into account the points mentioned in the preceding paragraph and considering that the described assignment procedure has been applied to the 10 data sets, it may be stated that there is at least one solution with a maximal fitness level for each of these data sets. Then, considering that a solution with a maximal fitness level outperforms all other possible solutions, this solution may be considered an optimal solution. Thus, it is possible to state that for each of the 10 designed data sets, there is at least one optimal solution with a fitness level equal to 9.

4.2. Evolutionary algorithm parameters

The evolutionary algorithm has four main parameters (i.e., size of initial population, crossover probability P_c, mutation probability P_m, and number of iterations). The effectiveness and efficiency of the algorithm greatly depend on the values assigned to these parameters. For this reason, preliminary experiments were conducted to determine what combination of values allows the algorithm to achieve the best results on the 10 data sets. These preliminary experiments are described below.

First, different values were defined for each parameter. Specifically, three values were considered for the initial population size (i.e., 50, 100 and 200). Then, different values were considered for P_c (i.e., values in the range [0.5,0.9]) and different values were considered for P_m (i.e., values in the range [0.01,0.1]). Finally, different values for the number of iterations were considered (i.e., 400, 600, and 1000). The combination of all these values yielded different possible settings for the evolutionary algorithm parameters.

The obtained settings were evaluated on the 10 data sets by means of the following procedure. Each setting was run 20 times on each of the 10 data sets. Then, in order to determine the effectiveness and efficiency of a given setting with respect to a given data set, we calculated the average fitness level of the 20 solutions obtained by means of the 20 runs, and we also calculated the average computation time of the 20 runs.

The procedure described above was utilized because the evolutionary algorithm is a stochastic method. In such methods, different runs of a given setting on a given data set may achieve different solutions. Therefore, a single run of a given setting on a given data set is not sufficient to establish the effectiveness and efficiency of this setting with respect to the data set. In this sense, it is necessary to perform several runs, and subsequently, to calculate the average fitness level of the solutions obtained by means of the runs. This average fitness value is considered a representative value of the effectiveness of the setting on the data set. Analogously, it is necessary to calculate the average computation time of the performed runs so as to obtain a representative value of the efficiency of the setting on the data set.

Table 3 presents the parameters' setting through which the evolutionary algorithm obtained the best results on the 10 data sets. This setting is considered in the following subsections.

4.3. Main results

The evolutionary algorithm was run 20 times on each of the 10 data sets. Then for each data set, the average fitness value of the obtained solutions and the average computation time of the runs were calculated. To perform these runs, the algorithm parameters were set with the values shown in Table 3.

Table 4 presents the results obtained by the algorithm on each data set. The first column provides the name of each data set; the second columns indicates the average fitness value of the achieved solutions for each data set; and the third column shows the average computation time of the runs performed on each data set.

Table 3
Best setting for the parameters of the evolutionary algorithm.

Parameter	Value
Crossover probability P_c	0.8
Mutation probability P_m	0.05
Population size	100
Number of generations	400

Table 4
Results obtained by the evolutionary algorithm.

Data set	Fitness value	Time (seconds)
1	9	0.5537
2	9	1.3741
3	9	11.0669
4	9	17.5976
5	8.8	40.8722
6	8.76	55.7548
7	8.7	196.9964
8	8.64	362.0328
9	8.61	574.6589
10	8.592	771.6553

In order to analyze the results presented in Table 4, it should be considered that, as mentioned in Subsection 4.1, each of the 10 data sets has at least one optimal solution with a fitness level equal to 9. Then, this fitness level is considered a reference level to evaluate the effectiveness of the evolutionary algorithm on each data set.

The results presented in Table 4 indicate that, for each of the first four data sets, the algorithm has achieved an optimal solution in each of the runs. Besides, for each of the six remaining data sets, the algorithm has achieved an average fitness value that is higher than 8.5. This means that, for the last six data sets, the algorithm has achieved solutions that are very near the optimal solutions. The composition of the obtained solutions for the last six data sets has been analyzed. On the basis of this analysis, it is possible to say that each of these solutions contains a high percentage of perfectly balanced teams.

Regarding the average computation time required by the algorithm, the following points may be mentioned. For each of the first six data sets, the average time required by the algorithm was lower than 60 s. Then, for each of the four remaining data sets, the average time required by algorithm was higher than 100 s and lower than 800 s. Taking into consideration the complexity of the addressed problems, particularly the complexity of the problems inherent in the last four data sets, the average computation times required by the algorithm on the 10 data sets are considered acceptable.

On the basis of these results, it may be said that for each of the 10 data sets, the algorithm has achieved high-quality solutions in an acceptable period of time.

4.4. Comparison with two competing methods

In this subsection, the performance of the evolutionary algorithm is compared with those of two competing algorithms, the exhaustive and random methods.

4.4.1. Comparison with the exhaustive method

In this subsection, the performance of the evolutionary algorithm is compared with that of another search method known as exhaustive method. In this comparison, the achieved performances are considered in terms of the average fitness value and the average computation time on each of the 10 data sets.

Unlike the evolutionary algorithm, the exhaustive method exhaustively enumerates all possible solutions to a given problem, which guarantees that the optimal solution will be found in each run. Therefore, the exhaustive method was run only once on each of the 10 data sets.

Table 5 shows the results obtained by the exhaustive method and those obtained by the evolutionary algorithm on the 10 data sets with respect to the average fitness value and the average computation time.

The results show that the exhaustive method was only able to solve the problems inherent in the first three data sets in a reasonable period of time. The reason for this is that the team-formation problem is an *NP-Hard* problem, so the computation time required by the exhaustive method increases exponentially as the problem size increases. Unlike the exhaustive method, the evolutionary algorithm has found optimal solutions for the first four data sets and near-optimal solutions for the remaining six data sets. The solutions for all 10 data sets have been found in an acceptable period of time. Furthermore, even though the computation time of the evolutionary algorithm also increased as the problem size increased, this increase is smaller than the one in the exhaustive

Table 5
Results obtained by the exhaustive method and results obtained by the evolutionary algorithm.

Data set	Exhaustive method		Evolutionary algorithm	
	Fitness value	Time (seconds)	Fitness value	Time (seconds)
1	9	59.4552	9	0.5537
2	9	189.27	9	1.3741
3	9	1072.587	9	11.0669
4	*N/A*	*N/A*	9	17.5976
5	*N/A*	*N/A*	8.8	40.8722
6	*N/A*	*N/A*	8.76	55.7548
7	*N/A*	*N/A*	8.7	196.9964
8	*N/A*	*N/A*	8.64	362.0328
9	*N/A*	*N/A*	8.61	574.6589
10	*N/A*	*N/A*	8.592	771.6553

method. Therefore, the evolutionary algorithm may be considered to obtain high-quality solutions efficiently.

4.4.2. Comparison with the random method

In this subsection, the performance of the evolutionary algorithm is compared with that of an algorithm known as random method. In this comparison, the achieved performances are considered in terms of the average fitness value and the average computation time on each of the 10 data sets.

The random method has been considered because it implements one of the grouping strategies that teachers most widely employ in their classes. This strategy is called random assignment strategy, and by means of its application, each of the n students in a class is randomly assigned to one of the g teams. The random method is a stochastic method. As a result, like the evolutionary algorithm, the random method was run 20 times on each of the 10 data sets. Then, the average fitness value of the obtained solutions and the average computation time of the runs were calculated for each data set.

Table 6 shows the results obtained by the random method and those obtained by the evolutionary algorithm on the 10 data sets with respect to the average fitness value and the average computation time.

The results indicate that the computation times required by the random method are much shorter than those required by the evolutionary algorithm. In addition, it may be seen that the time required by the random method was not significantly affected by the complexity of the addressed problems. Conversely, the time required by the evolutionary algorithm increased as the problem complexity increased. Although the evolutionary algorithm required a longer computation time than the random method, when one observes the average fitness values achieved by the two algorithms, it is possible to state that the quality of the solutions found by the evolutionary algorithm is much greater than the quality of the solutions found by the random method. Specifically, the average fitness values obtained by the evolutionary algorithm are higher than 8.5, whereas those obtained by the random method are lower than 2. Furthermore, even though the average fitness values obtained by the evolutionary algorithm decrease as the problem complexity increases, such decrease is very small compared with the decrease in the random method. Therefore, the evolutionary algorithm may be considered to obtain higher-quality solutions than the random method.

Table 6
Results obtained by the random method and results obtained by the evolutionary algorithm.

Data set	Random method		Evolutionary algorithm	
	Fitness value	Time (seconds)	Fitness value	Time (seconds)
1	1.5	0.0001	9	0.5537
2	1.3	0.0002	9	1.3741
3	−1.2	0.0014	9	11.0669
4	−1.8	0.0022	9	17.5976
5	−3.1	0.0051	8.8	40.8722
6	−3.9	0.0069	8.76	55.7548
7	−4.5	0.0246	8.7	196.9964
8	−5.2	0.0452	8.64	362.0328
9	−6.1	0.0718	8.61	574.6589
10	−7.3	0.0964	8.592	771.6553

5. Conclusions

This paper has addressed the problem of forming collaborative learning teams from a students' class. As part of the problem, we considered a grouping criterion that is widely used by teachers in their classrooms. This criterion is based on taking into account the students' roles and on forming well-balanced teams according to the roles of their members. This criterion has been implemented following the team role model proposed by Belbin (Belbin, 1981, 1993) and the balance conditions established by this author.

To solve the addressed problem, we propose a deterministic crowding evolutionary algorithm. Taking into account a given students' class that must be divided into a given number of teams, the algorithm designs different alternatives to divide the students into learning teams and evaluates each alternative with respect to the grouping criterion considered as part of the problem. This evaluation is conducted based on knowledge of the students' roles.

Different computational experiments were developed for evaluating the evolutionary algorithm's performance. These experiments involved solving team formation problems inherent in ten data sets with different levels of complexity. Then, the performance of the algorithm on each data set was determined by calculating the average fitness value of the obtained solutions as well as the average computation time of the runs. Additionally, the performance of the evolutionary algorithm on the ten data sets was compared with the performances of the exhaustive and random methods on the same data sets.

On the basis of the results obtained by the evolutionary algorithm on the ten data sets, it may be stated that this algorithm has been able to achieve high-quality solutions in an acceptable computation time for each of the data sets employed. Furthermore, as a result of the comparative analysis conducted, it may be said that the evolutionary algorithm has been shown to have advantages over the exhaustive and random methods. Firstly, the evolutionary algorithm has been able to effectively solve all those problems for which the exhaustive method has not found a solution in an acceptable period of time. Secondly, the evolutionary algorithm has been shown to be more effective than the random method on each of the ten data sets. Considering all the observations hereby mentioned, we conclude that the results achieved by the evolutionary algorithm are really promising.

References

Alfonseca, E., Carro, R. M., Martín, E., Ortigosa, A., & Paredes, P. (2006). The impact of learning styles on student grouping for collaborative learning: A case study. *User Modeling and User-Adapted Interaction, 16*(3–4), 377–401.

Barkley, E. F., Cross, K. P., & Howell Major, C. (2005). *Collaborative learning techniques*. John Wiley & Sons, Inc.

Beane, W. E., & Lemke, E. A. (1971). Group variables influencing the transfer of conceptual behavior. *Journal of Educational Psychology, 62*(3), 215–218.

Belbin, R. M. (1981). *Management teams: Why they succeed or fail*. Oxford: Butterworth-Heinemann.

Belbin, R. M. (1993). *Team roles at work*. Oxford: Butterworth-Heinemann.

Cameron, S. (2002). *Business student's handbook: Learning skills for study and employment*. Harlow: Prentice Hall.

Dalton, D. W., Hannafin, M. J., & Hooper, S. (1989). Effects of individual and cooperative computer-assisted instruction on student performance and attitudes. *Educational Technology Research and Development, 37*(2), 15–24.

Davis, J., Millburn, P., Murphy, T., & Woodhouse, M. (1992). *Successful team building: How to create teams that really work*. London: Kogan Page.

Eiben, A. E., & Smith, J. E. (2007). *Introduction to evolutionary computing* (2nd ed.). Springer. ISBN: 978-3-540-40184-1.

Goldberg, D. E. (2007). *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Publishing Company, Inc..

Hooper, S., & Hannafin, M. J. (1988). Cooperative CBI: The effects of heterogeneous versus homogeneous group on the learning of progressively complex concepts. *Journal of Educational Computing Research, 4*(4), 413–424.

Jeffries, P., Grodzinsky, F., & Griffin, J. (2003). Advantages and problems in using information communication technologies to support the teaching of a multi-institutional computer ethics course. *Journal of Educational Media, 28*(2–3), 191–202.

Jeffries, P., Grodzinsky, F., & Griffin, J. (2004). Building successful on-line learning communities across international boundaries: A case study. *Proceedings of ETHICOMP 2004 (The Seventh ETHICOMP international conference on the social and ethical impacts of information and communication technologies), 14–16 April, 2004.*. Syros, Greece: University of the Aegean.

Johansen, T. (2003). Predicting a Team's Behaviour by Using Belbin's Team Role Self Perception Inventory. Ph.D. thesis, University of Stirling, 2003.

Lin, Y. T., Huang, Y. M., & Cheng, S. C. (2010). An automatic group composition system for composing collaborative learning groups using enhanced particle swarm optimization. *Computers and Education, 55*, 1483–1493.

Margerison, C., & McCann, D. (1990). *Team management*. London: W.H. Allan.

Martin, E., & Paredes, P. (2004). Using learning styles for dynamic group formation in adaptive collaborative hypermedia systems. In *Proceedings of workshops in connection with 4th international conference on web engineering* (pp. 188–197). Munich: Rinton Press, Inc.

McFadzean, E. (2001). Supporting virtual learning groups. Part 2: An integrated approach. *Team Performance Management: An International Journal, 7*(5/6), 77–92.

Meyer, D. (2009). Optassign – A web-based tool for assigning students to groups. *Computers and Education, 53*(4), 1104–1119.

Michaelsen, L. K., Knight, A. B., & Fink, L. D. (2004). *Team-based learning: A transformative use of small groups in college teaching*. Sterling, VA: Stylus Publishing.

Nielsen, T., Hvas, A. E., & Kjaergaard, A. (2009). Student team formation based on learning styles at university start: Does it make a difference to the student? *Reflection Education, 5*(2), 85–103.

Ounnas, A. (2010). Enhancing the Automation of Forming Groups for Education with Semantics. PhD Thesis, Faculty of Engineering and Applied Science, Department of Electronics and Computer Science, University of Southampton, November 2010.

Park, W., & Bang, H. (2002). Team role balance and team performance. In *Belbin Biennial conference, changing role of management in the 21st century*. Cambridge: Clare College.

Parker, G. M. (1990). *Team players and teamwork: The competitive business strategy*. Oxford: Jossey-Bass.

Prichard, J. S., & Stanton, N. A. (1999). Testing Belbin's team role theory of effective groups. *The Journal of Management Development, 18*(8), 652–660.

Rutherfoord, R. H. (2001). Using personality inventories to help form teams for software engineering class projects. In *ITiCSE '01: Proceedings of the 6th annual conference on innovation and technology in computer science education* (pp. 73–76). New York, NY, USA: ACM.

Saleh, I., & Kim, S. (2009). A fuzzy system for evaluating students' learning achievement. *Expert Systems with Applications, 36*(3), 3243–6236.

Sánchez Hórreo, V., & Carro, R. M. (2007). Studying the impact of personality and group formation on learner performance. In *Groupware: Design, implementation, and use*. *LNCS* (Vol. 4715, pp. 287–294). Springer-Verlag. ISSN: 0302-9743.

Sommerville, J., & Dalziel, S. (1998). Project teambuilding – The applicability of Belbin's team-role self-perception inventory. *International Journal of Project Management, 16*(3), 165–171.

Speck, B. W. (2003). Fostering collaboration among students in problem-based learning. *New Direct. Teach. Learn, 95*, 59–66.

Spencer, J., & Pruss, A. (1992). *Managing your team*. London: Piatkus.

Stevens, K. (1998). The Effects of Roles and Personality Characteristics on Software Development Team Effectiveness. Ph.D thesis, Faculty of Virginia Polytechnic Institute and State University, 1998.

Tang, T., Chan, K., Winoto, P., & Wu, A. (2001). Forming student clusters based on their browsing behaviors. In *Proceedings of the 9th international conference on computers in education (ICCE 2001) Seoul, Korea* (pp. 1229–1235).

Webb, N. M. (1982). Student Interaction and learning in small groups. *Review of Educational Research, 52*(3), 421–455.

Wilkinsona, I. A. G., & Fung, I. Y. Y. (2002). Small-group composition and peer effects. *International Journal of Educational Research, 37*(5), 425–447.

Winter, M. (2004). Developing a group model for student software engineering teams. Master's thesis, University of Saskatchewan, 2004.

Woodcock, M. (1989). *Team development manual*. Aldershot: Gower.

Yang, S. J. H. (2006). Context aware ubiquitous learning environments for peer-to-peer collaborative learning. *Journal of Educational Technology and Society, 9*(1), 188–201.