A Hybrid Evolutionary Algorithm based on Adaptive Mutation and Crossover for Collaborative Learning Team Formation in Higher Education

Virginia Yannibelli[1,2(✉)] and Analía Amandi[1,2]

[1] ISISTAN Research Institute, UNCPBA University, Campus Universitario, Paraje Arroyo Seco, 7000 Tandil, Argentina
{vyannibe,amandi}@exa.unicen.edu.ar
[2] CONICET, National Council of Scientific and Technological Research, Buenos Aires, Argentina

Abstract. In this paper, we address a collaborative learning team formation problem in higher education environments. This problem considers a grouping criterion successfully evaluated in a wide variety of higher education courses and training programs. To solve the problem, we propose a hybrid evolutionary algorithm based on adaptive mutation and crossover processes. The behavior of these processes is adaptive according to the diversity of the evolutionary algorithm population. These processes are meant to enhance the evolutionary search. The performance of the hybrid evolutionary algorithm is evaluated on ten different data sets, and then, is compared with that of the best algorithm previously proposed in the literature for the addressed problem. The obtained results indicate that the hybrid evolutionary algorithm considerably outperforms the previous algorithm.

Keywords: Collaborative learning · Collaborative learning team formation · Team roles · Evolutionary algorithms · Hybrid evolutionary algorithms · Adaptive evolutionary algorithms · Simulated annealing algorithms

1 Introduction

In higher education environments, collaborative learning is a pedagogical approach usually used to both complement and enrich the individual learning of students. This approach requires organizing students into collaborative learning teams. The students of each collaborative learning team must work together to achieve shared learning goals. The collaborative learning teams must be formed so that students can acquire new knowledge and skills through the interaction with their peers, improving their individual learning. In this context, the grouping criterion (i.e., the criterion to form collaborative learning teams) is highly relevant because of the composition of each collaborative learning team affects the learning level and the social behavior of their student members as well as the performance of the team [1, 2]. Moreover, the way in which the grouping criterion is applied (i.e., either manually or automatically) is very

© Springer International Publishing AG 2017
H. Yin et al. (Eds.): IDEAL 2017, LNCS 10585, pp. 345–354, 2017.
https://doi.org/10.1007/978-3-319-68935-7_38

relevant because of many known grouping criteria require a considerable amount of knowledge, time and effort to be manually applied [10]. In such cases, it is possible to reduce considerably the workload of professors and also optimize the collaborative learning team formation through automation.

Different works in the literature have described and addressed the problem of forming collaborative learning teams automatically from the students [4, 10]. These works differ in relation to several aspects including the grouping criteria considered, and the algorithms utilized. In this regards, to the best of our knowledge, only few of these works consider grouping criteria that have been both successfully and widely evaluated in higher education environments.

In [5], the authors describe the problem of forming collaborative learning teams automatically from the students enrolled in a given course. As part of the problem, the authors consider a grouping criterion successfully evaluated in a wide variety of higher education courses and training programs. Such grouping criterion corresponds to the criterion defined by Belbin's team role model [3]. This criterion considers the team roles of students, and implies forming well-balanced teams regarding the team roles of their members. A team role is the way in which a person tends to behave, contribute and interrelate with others throughout a collaborative task. In this respect, the Belbin's model [3] defines nine team roles and balance conditions. Many different studies in the literature indicate that collaborative learning team formation in higher education environments according to the Belbin's criterion leads to good interactions and discussions 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 teams [4]. Thus, it is considered that the collaborative learning team formation problem described in [5] is really valuable in the context of higher education environments.

In this paper, we present a hybrid evolutionary algorithm to solve the collaborative learning team formation problem described in [5]. This algorithm utilizes adaptive mutation and crossover processes that adapt their behavior according to the diversity of the evolutionary algorithm population. The utilization of these adaptive processes is meant to improve the evolutionary search performance [6, 12, 13].

We present this hybrid evolutionary algorithm because of the following reasons. The collaborative learning team formation problem described in [5] is an NP-Hard optimization problem. In this respect, evolutionary algorithms with adaptive mutation and crossover processes have been proven to be more effective than evolutionary algorithms with non-adaptive mutation and crossover processes in the resolution of a wide variety of NP-Hard optimization problems [6, 12, 13]. Therefore, we consider that the hybrid evolutionary algorithm presented could outperform the best algorithm previously presented in the literature for the addressed problem. We refer to the hybrid evolutionary algorithm presented in [8].

The remainder of the paper is organized as follows. In Sect. 2, we describe the problem addressed. In Sect. 3, we present the hybrid evolutionary algorithm. In Sect. 4, we present the computational experiments carried out to evaluate the performance of the hybrid evolutionary algorithm and an analysis of the results obtained. In Sect. 5, we present related works. Finally, in Sect. 6 we present the conclusions of the present work.

2 Problem Description

In this paper, we address the collaborative learning team formation problem described in [5]. We present below a description of this problem.

Suppose a course S has n students enrolled, $S = \{s_1, s_2, \ldots, s_n\}$, and the professor must organize the n students into g teams, $G = \{G_1, G_2, \ldots, G_g\}$. Each G_i team is composed of a z_i number of students, and each student can only belong to one team. Regarding team size, students must be organized so that the g teams have a similar number of students each. Specifically, the difference among the sizes of the teams must not exceed one. The values of the terms S, n and g are known.

As regards the students, it is considered that they naturally play different team roles when participating in a collaborative task. Regarding the team roles that can be played by the students, the nine team roles defined in Belbin's model [3] are considered. Table 1 presents these nine roles and a brief description of the features of each.

According to Belbin's model [3], it is considered that each student naturally plays one or several of the nine roles presented in Table 1. In this sense, the roles naturally played by each student are known data. These roles may be obtained through the Belbin Team-Role Self-Perception Inventory (BTRSPI) developed by Belbin [3].

As part of the problem, teams must be composed so that the balance among the team roles of their members is maximized. This grouping criterion requires analyzing the balance level of the formed teams. To analyze such level, the balance conditions established by Belbin are considered [3]. Regarding these conditions, Belbin [3] 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 [3]. 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) [3].

Table 1. Belbin's role characteristics.

Role	Characteristics
Plant (PL)	Creative, imaginative, unorthodox. Solves difficult problems.
Resource Investigator (RI)	Extrovert, enthusiastic, communicative. Explore opportunities. Develops contacts.
Co-ordinator (CO)	Mature, confident, a good chairperson. Clarifies goals, promotes decision-making, delegates well.
Shaper (SH)	Challenging, dynamic, thrives on pressure. Has the drive and courage to overcome obstacles.
Monitor Evaluator (ME)	Sober, strategic and discerning. Sees all options. Judges accurately.
Teamworker (TW)	Co-operative, mild, perceptive and diplomatic. Listens, builds, averts friction.
Implementer (IM)	Disciplined, reliable, conservative and efficient. Turns ideas into practical actions.
Completer/Finisher (CF)	Painstaking, conscientious, anxious. Searches out errors and omissions. Polishes and perfects.
Specialist (SP)	Single-minded, self-starting, dedicated. Provides skills in key areas.

The grouping criterion considered as part of the problem is modeled by Formulas (1), (2) and (3). Formulas (1) and (2) model the balance conditions defined by Belbin [3]. Formula (1) analyzes the way in which a given r role is played within a given G_i team and then gives a score accordingly. When r is naturally played by only one member of G_i team, then 1 point is awarded to G_i. Conversely, when 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) defines the balance level of a given G_i team. This level is established based on the scores obtained by G_i, through Formula (1), regarding the nine roles. Thus, the greater the number of non-duplicate roles (i.e., roles played 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 defined by Belbin [3] can be seen in Formula (2). By using this formula, a perfectly balanced team (i.e., a team in which each of the nine roles is played naturally by only one team member) will obtain a level equal to 9.

Formula (3) maximizes the average balance level of g teams defined from the n students of the course. In other words, this formula aims 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 addressed problem. In Formula (3), 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 $b(G)$ represents the average balance level of the g teams belonging to set G. Then, Formula (3) utilizes Formula (2) to define the balance level of each G_i team belonging to the G set. Note that in the case of a G set of perfectly balanced g teams, the value of the term $b(G)$ is equal to 9.

For a more detailed discussion of Formulas (1), (2) and (3), we refer to [5].

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

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

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

3 Hybrid Evolutionary Algorithm

The general behavior of the hybrid evolutionary algorithm proposed for the addressed problem is described as follows. Considering a course with n students who shall be organized into g teams, the algorithm starts creating a random initial population of feasible encoded solutions. In this population, each solution codifies a feasible set of g teams which may be defined from the n students. To encode these solutions, the representation proposed in [5] is used. Then, the algorithm decodes and evaluates each solution of the population by a fitness function. The set of g teams inherent to each solution is built by the decoding process proposed in [5], and then evaluated regarding the optimization objective of the problem. As mentioned in Sect. 2, this objective is maximizing the balance level of the g teams formed from n students. Thus, the fitness function evaluates the balance level of the g teams represented by each solution and defines a fitness level for each solution (i.e., the fitness function calculates the value of the term $b(G)$ corresponding to each solution by Formulas (3), (2) and (1)). To such evaluation, the function uses knowledge of the students' roles.

Once each solution of the population is evaluated, a well-known parent selection process named roulette wheel selection process [6] is used to decide which solutions of the population will compose the mating pool. By this process, the highest fitness solutions will have more probability of being selected for the mating pool. After the mating pool is complete, the solutions in the mating pool are paired. Then, a crossover process named partially mapped crossover [6] is applied to each of these pairs of solutions with an adaptive probability AP_c, to generate new feasible ones. Then, a mutation process named insert mutation [6] is applied to each solution obtained by the crossover process, with an adaptive probability AP_m. Then, the traditional fitness-based steady-state selection process [6] is applied in order to define which solutions from the solutions in the population and the solutions generated from the mating pool will integrate the new population. This survival selection process preserves the best solutions found by the hybrid evolutionary algorithm [6]. Finally, an adaptive simulated annealing algorithm is applied to each solution of the new population, excepting the best solution which is preserved.

This process is repeated until a predefined number of iterations is reached.

3.1 Adaptive Mutation and Adaptive Crossover

The above-mentioned crossover and mutation processes are applied with adaptive crossover and mutation probabilities, respectively. In this respect, we defined the adaptive crossover probability AP_c and the adaptive mutation probability AP_m. These probabilities are defined by Formulas (4) and (5), where PD refers to the population diversity, and PD_{MAX} refers to the maximum PD attainable. In Formula (4), the terms C^H and C^L represent to the upper and lower bounds for the crossover probability, respectively. In Formula (5), the terms M^H and M^L represent to the upper and lower bounds for the mutation probability, respectively. Then, f_{max} is the maximal fitness of the population, f_{min} is the minimal fitness of the population, and f is the fitness of the solution to be mutated.

The term PD is defined by Formula (6), where f_{max} is the maximal fitness of the population, f_{avg} is the average fitness of the population, and $(f_{max} - f_{avg})$ is a measure of the population diversity. This measure has been proposed by Srinivas and Patnaik [7], and is one of the population diversity measures most well-known in the literature [6].

The term PD_{MAX} is defined by Formula (7), where f_{MAX} and f_{MIN} correspond to the upper and lower bounds for the fitness function, respectively.

By Formulas (4)–(7), AP_c and AP_m are adaptive regarding the population diversity. When the population diversity decreases, AP_c and AP_m are increased, promoting the exploration of unvisited regions of the search space. This is important to prevent the premature convergence of the evolutionary search. When the population is diverse, AP_c and AP_m are decreased, promoting the exploitation of visited regions of the search space.

By Formula (5), AP_m is also adaptive according to the fitness of the solution to be mutated. In this regards, lower values of AP_m are defined for high-fitness solutions, and higher values of AP_m are defined for low-fitness solutions. This is meant in order to preserve high-fitness solutions, while disrupting low-fitness solutions to promote the exploration of the search space.

$$AP_c = \left(\frac{PD_{MAX} - PD}{PD_{MAX}}\right) * \left(C^H - C^L\right) + C^L \tag{4}$$

$$AP_m = \left(\frac{f_{max} - f}{f_{max} - f_{min}}\right) * \left(\frac{PD_{MAX} - PD}{PD_{MAX}}\right) * \left(M^H - M^L\right) + M^L \tag{5}$$

$$PD = (f_{max} - f_{avg}) \tag{6}$$

$$PD_{MAX} = (f_{MAX} - f_{MIN}) \tag{7}$$

3.2 Adaptive Simulated Annealing Algorithm

The adaptive simulated annealing algorithm applied is a variant of the one proposed in [8], and is described below.

The adaptive simulated annealing algorithm is an iterative process. This process starts from a given encoded solution s and a given initial value T_0 for the temperature parameter. In each iteration, the algorithm generates a new encoded solution s' from the current encoded solution s by a move operator, and then decides if s should be replaced by s'. If the fitness value of s' is higher than that of s, the algorithm replaces to s by s'. Otherwise, if the fitness value of s' is not higher than that of s, the algorithm replaces to s by s' with an acceptance probability equal to $exp(-\Delta/T)$, where T is the temperature current value, and Δ is the difference between the fitness values of s and s'. This probability mainly depends on T. When T is high, the probability is also high, and vice versa. T is reduced by a given cooling factor at the end of each iteration. This process is repeated until a predefined number of iterations is reached.

Regarding the initial value T_0 for the temperature parameter, we defined this value according to the population diversity. Specifically, T_0 is inversely proportional to the

population diversity, and is calculated by the next formula: $T_0 = 1/PD$, where PD refers to the population diversity, as mentioned in Sect. 3.1. By this formula, when the population is diverse, T_0 is low, and therefore, the acceptance probability of the simulated annealing algorithm is also low. As consequence of this, the algorithm fine-tunes the solutions of the population, promoting the exploitation of visited regions of the search space. When the population diversity decreases, T_0 increases, and therefore, the acceptance probability of the simulated annealing algorithm also increases. As consequence of this, the algorithm introduces diversity into the population, promoting the exploration of unvisited regions of the search space. Thus, the algorithm is adaptive to promote either the exploitation or exploration of the search space.

Regarding the move operator of the simulated annealing algorithm, we applied a well-known operator named swap mutation [6].

4 Computational Experiments

We used the ten data sets introduced in [5] to evaluate the performance of the hybrid evolutionary algorithm. Table 2 presents the main characteristics of these data sets. For a detailed description of the team roles of the students in each data set, we refer to [5]. Each data set has a known optimal solution with a fitness level equal to 9. These optimal solutions are considered here as references.

Table 2. Main characteristics of the data sets.

Data set	Number of students (n)	Number of teams to be built (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

We run the algorithm 30 times on each data set. After each run, this algorithm provided the best solution achieved. To develop these runs, we set the algorithm parameters as follows: population size = 80; number of iterations = 200; crossover process: $C^H = 0.9$ and $C^L = 0.5$; mutation process: $M^H = 0.2$ and $M^L = 0.01$; survival selection process: replacement factor = 40; simulated annealing algorithm: number of iterations = 20 and cooling factor = 0.9. The algorithm parameters were set based on preliminary experiments that showed that these values led to the best and most stable results.

We analyzed the results obtained by the hybrid evolutionary algorithm for each of the ten data sets. Specifically, we analyzed the average fitness value of the solutions reached for each data set, and 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 3 GB RAM under Windows XP Professional Version 2002. The algorithm was implemented in Java.

For the data sets 1–7 (i.e., the seven less complex data sets), the algorithm reached an average fitness value equal to 9. This means that the algorithm achieved an optimal solution in each run. For the data sets 8–10 (i.e., the three more complex data sets), the algorithm reached average fitness values equal to 8.87, 8.81 and 8.76, respectively. This means that the algorithm reached very near-optimal solutions for each data set.

Regarding the time required by the algorithm, we may mention the following. For the data sets 1–6 (i.e., the six less complex data sets), the average time required was 0.18, 0.46, 3.78, 6.01, 14.9 and 19.03 seconds, respectively. For the data sets 7–10 (i.e., the four more complex data sets), the average time required was 72.4, 133.07, 211.18 and 303.84 seconds, respectively. Considering the complexity of the problem instances represented by the data sets, the average computation times required by the algorithm are considered acceptable.

4.1 Comparison with a Competing Algorithm

In this section, we compare the performance of the hybrid evolutionary algorithm with that of the best algorithm previously presented in the literature for the addressed problem. We refer to the hybrid evolutionary algorithm presented in [8].

For simplicity, we will refer to the hybrid evolutionary algorithm presented in [8] as algorithm H. Like the hybrid evolutionary algorithm presented here, the algorithm H incorporates an adaptive simulated annealing algorithm within the framework of an evolutionary algorithm. Unlike the hybrid evolutionary algorithm presented here, the algorithm H uses non-adaptive crossover and mutation processes. These processes do not consider the population diversity.

In the experiments reported in [8], the algorithm H has been evaluated on the ten data sets presented in Table 2, and has obtained the results that are mentioned below. These experiments were performed on a personal computer Intel Core 2 Duo at 3.00 GHz and 3 GB RAM under Windows XP Professional Version 2002. The algorithm was implemented in Java.

For the data sets 1–5, the algorithm H obtained an average fitness value equal to 9. For the data sets 6–10, the algorithm H obtained average fitness values equal to 8.97, 8.86, 8.77, 8.74 and 8.7, respectively. In relation to the time required by the algorithm H, for the data sets 1–6, the average time required was 0.29, 0.721, 5.81, 9.24, 21.46 and 29.27 seconds, respectively. For the data sets 7–10, the average time required was 103.43, 190.1, 301.69 and 405.118 seconds, respectively.

Comparing the results obtained by the algorithm H and the hybrid evolutionary algorithm proposed here, we can mention the following points. Both algorithms have obtained an optimal average fitness value for the data sets 1–5 (i.e., the less complex data sets). However, the average fitness value obtained by the hybrid evolutionary algorithm for the data sets 6–10 (i.e., the five more complex data sets) is significantly

higher than that obtained by the algorithm H. Besides, the average time required by the hybrid evolutionary algorithm for each one of the data sets is much lower than that required by the algorithm H.

Therefore, the hybrid evolutionary algorithm outperforms the algorithm H on the more complex data sets. The main reason for this is that, unlike the algorithm H, the hybrid evolutionary algorithm utilizes adaptive mutation and crossover processes. These processes adapt their behavior according to the population diversity, to promote either the exploration or exploitation of the search space, and therefore, enhance the performance of the evolutionary search. Thus, the hybrid evolutionary algorithm can reach better solutions in less computation time than algorithm H on the more complex data sets.

5 Related Works

To the best of our knowledge, only few works in the literature address the problem of automatically forming collaborative learning teams based on the Belbin's model [4, 10]. These works differ mainly in relation to the modeling of this problem, and the algorithms proposed to solve it.

In the framework proposed in [11], this team formation problem is modeled as a constraint satisfaction problem, and is solved by a DLV constraint satisfaction solver. In the tool proposed in [4], this team formation problem is modeled as a coalition structure generation problem, and is solved by means a linear programming method.

The two above-mentioned works propose exhaustive search algorithms to solve the problem. However, this kind of algorithms only can solve very small instances of the problem in a reasonable period of time.

In [5, 8, 9], different evolutionary algorithms are presented with the aim of solving problem instances with very different complexity levels. Such algorithms, particularly the algorithm presented in [8], achieved promising results. However, these algorithms use non-adaptive mutation and crossover processes for developing the evolutionary search.

6 Conclusions

In this paper, we proposed a hybrid evolutionary algorithm to solve the collaborative learning team formation problem described in [5]. This algorithm utilizes adaptive mutation and crossover processes, to improve the evolutionary search performance. The behavior of such processes is adaptive in order to promote either exploration or exploitation of the search space, according to the population diversity. The presented computational experiments show that the hybrid evolutionary algorithm significantly outperforms the best algorithm previously proposed in the literature for solving the addressed problem.

In future works, we will evaluate other adaptive mutation and crossover processes. Besides, we will evaluate adaptive parent selection and survival selection processes. Moreover, we will evaluate the integration of other adaptive search and optimization techniques into the framework of the evolutionary algorithm.

References

1. Barkley, E.F., Cross, K.P., Howell Major, C.: Collaborative Learning Techniques. Wiley, New York (2005)
2. Michaelsen, L.K., Knight, A.B., Fink, L.D.: Team-Based Learning: A Transformative Use of Small Groups in College Teaching. Stylus Publishing, Sterling (2004)
3. Belbin, R.M.: Team Roles at Work, 2nd edn. Taylor & Francis, London (2011)
4. Alberola, J., Del Val, E., Sanchez-Anguix, V., Palomares, A., Teruel, M.: An artificial intelligence tool for heterogeneous team formation in the classroom. Knowl.-Based Syst. **101**(1), 1–14 (2016)
5. Yannibelli, V., Amandi, A.: A deterministic crowding evolutionary algorithm to form learning teams in a collaborative learning context. Expert Syst. Appl. **39**(10), 8584–8592 (2012)
6. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer, Heidelberg (2015)
7. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994)
8. Yannibelli, V., Amandi, A.: A Hybrid Algorithm Combining an Evolutionary Algorithm and a Simulated Annealing Algorithm to Solve a Collaborative Learning Team Building Problem. In: Pan, J.-S., Polycarpou, Marios M., Woźniak, M., de Carvalho, A.C.P.L.F., Quintián, H., Corchado, E. (eds.) HAIS 2013. LNCS, vol. 8073, pp. 376–389. Springer, Heidelberg (2013). doi:10.1007/978-3-642-40846-5_38
9. Yannibelli, V., Amandi, A.: A Memetic Algorithm for Collaborative Learning Team Formation in the Context of Software Engineering Courses. In: Cipolla-Ficarra, F., Veltman, K., Verber, D., Cipolla-Ficarra, M., Kammüller, F. (eds.) ADNTIIC 2011. LNCS, vol. 7547, pp. 92–103. Springer, Heidelberg (2012). doi:10.1007/978-3-642-34010-9_9
10. Cruz, W.M., Isotani, S.: Group Formation Algorithms in Collaborative Learning Contexts: A Systematic Mapping of the Literature. In: Baloian, N., Burstein, F., Ogata, H., Santoro, F., Zurita, G. (eds.) CRIWG 2014. LNCS, vol. 8658, pp. 199–214. Springer, Cham (2014). doi:10.1007/978-3-319-10166-8_18
11. Ounnas, A., Davis, H.C., Millard, D.E.: A framework for semantic group formation in education. Educational Tech. Soc. **12**(4), 43–55 (2009)
12. Rodriguez, F.J., García-Martínez, C., Lozano, M.: Hybrid metaheuristics based on evolutionary algorithms and simulated annealing: taxonomy, comparison, and synergy test. IEEE Trans. Evol. Comput. **16**(6), 787–800 (2012)
13. Talbi, E. (ed.): Hybrid Metaheuristics. SCI, vol. 434. Springer, Heidelberg (2013)