Scheduling Projects by a Hybrid Evolutionary Algorithm with Self-Adaptive Processes

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 present a hybrid evolutionary algorithm with self-adaptive processes to solve a known project scheduling problem. This problem takes into consideration an optimization objective priority for project managers: to maximize the effectiveness of the sets of human resources assigned to the project activities. The hybrid evolutionary algorithm integrates self-adaptive processes with the aim of enhancing the evolutionary search. The behavior of these processes is self-adaptive according to the state of the evolutionary search. The performance of the hybrid evolutionary algorithm is evaluated on six different instance sets and then is compared with that of the best algorithm previously proposed in the literature for the addressed problem. The obtained results show that the hybrid evolutionary algorithm considerably outperforms the previous algorithm.

Keywords: Project scheduling · Human resource assignment · Multi-skilled resources · Hybrid evolutionary algorithms · Evolutionary algorithms · Simulated annealing algorithms

1 Introduction

Project scheduling is a really central, complex and costly task in most organizations and companies. This task means defining start times and human resource assignments feasible for the activities of a given project, in such a way that a given optimization objective is achieved. In addition, defining human resource assignments for project activities implies considering the available knowledge about the effectiveness of the human resources with regard to project activities. This is because the results as well as the development of any project activity depend on the effectiveness of the human resource assignment defined for it [1, 2].

Based on the above-mentioned, a wide variety of project scheduling problems have been presented and addressed in the literature. Nonetheless, only few of these project scheduling problems consider that human resources usually have very different levels of effectiveness [3–6, 10], a really essential aspect in the context of real project scheduling problems. These project scheduling problems differ significantly in the assumptions considered regarding the effectiveness of human resources.

G. Sidorov and S.N. Galicia-Haro (Eds.): MICAI 2015, Part I, LNAI 9413, pp. 401–412, 2015.
DOI: 10.1007/978-3-319-27060-9_33

In the project scheduling problem formally described in [6], it is supposed that the effectiveness level of a human resource depends on different factors inherent to its work context (i.e., the project activity to which the resource is assigned, the skill to which the resource is assigned within the project activity, the set of human resources assigned to the project activity, and the attributes of the resource). This assumption about the effectiveness of human resources is really valuable. This is mainly because, in the context of real project scheduling problems, human resources have different levels of effectiveness with regards to different work contexts, and therefore, the effectiveness level of a human resource is usually considered in relation to its work context [1, 2]. To the best of our knowledge, the influence of the factors inherent to the work context on the effectiveness levels of human resources is not taken into account in other project scheduling problems described in the literature. Therefore, the project scheduling problem described in [6] supposes valuable and also novel assumptions regarding the effectiveness levels of human resources in the context of project scheduling problems.

In this paper, we present a hybrid evolutionary algorithm with self-adaptive processes to solve the project scheduling problem described in [6]. It is necessary to mention that this problem takes into account an optimization objective priority for project managers: to maximize the effectiveness levels of the sets of human resources assigned to the project activities. This algorithm integrates self-adaptive processes which adapt their behavior in accordance with the state of the evolutionary search. The integration of self-adaptive processes has the aim of enhancing the evolutionary search, in both exploration and exploitation [18–20].

We present this hybrid evolutionary algorithm mainly due to the next reasons. The addressed project scheduling problem is a special case of the RCPSP (Resource Constrained Project Scheduling Problem) [9], and therefore, is an NP-Hard problem. In this respect, hybrid evolutionary algorithms that integrate self-adaptive processes have been shown to be much more effective than hybrid evolutionary algorithms with non-adaptive processes in the resolution of a wide variety of NP-Hard problems [18–20]. Thus, 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 present a description of the addressed problem. In Sect. 3, we present the hybrid evolutionary algorithm. In Sect. 4, we present the computational experiments developed in order to evaluate the performance of the hybrid evolutionary algorithm and also an analysis of the obtained results. In Sect. 5, we review reported project scheduling problems in which the effectiveness of human resources is considered. Finally, in Sect. 6 we present the conclusions of the present work.

2 Problem Description

In this paper, we address the project scheduling problem formally presented in [6]. We give a description of this project scheduling problem below.

Consider that a project contains a set A of N activities, $A = \{1, \ldots, N\}$, which have to be scheduled (i.e., the starting time and the human resources of each activity have to

be defined). The duration, precedence relations and resource requirements of each activity are known.

The duration of each activity j is notated as d_j. Moreover, it is considered that pre-emption of activities is not allowed (i.e., the d_j periods of time must be consecutive).

Among some project activities, there are precedence relations. The precedence relations establish that each activity j cannot start until all its immediate predecessors, given by the set P_j, have completely finished.

Project activities require human resources – employees – skilled in different knowledge areas. Specifically, each activity requires one or several skills as well as a given number of employees for each skill.

It is considered that organizations and companies have a qualified workforce to develop their projects. This workforce is made up of a number of employees, and each employee masters one or several skills.

Considering a given project, set SK represents the K skills required to develop the project, $SK = \{1,\ldots, K\}$, and set AR_k represents the available employees with skill k. Then, the term $r_{j,k}$ represents the number of employees with skill k required for activity j of the project. The values of the terms $r_{j,k}$ are known for each project activity.

It is considered that an employee cannot take over more than one skill within a given activity. In addition, an employee cannot be assigned more than one activity at the same time.

Based on the previous assumptions, an employee can be assigned different activities but not at the same time, can take over different skills required for an activity but not simultaneously, and can belong to different possible sets of employees for each activity.

As a result, it is possible to define different work contexts for each available employee. It is considered that the work context of an employee r, denoted as $C_{r,j,k,g}$, is made up of four main components. The first component refers to the activity j which r is assigned (i.e., the complexity of j, its domain, etc.). The second component refers to the skill k which r is assigned within activity j (i.e., the tasks associated to k within j). The third component is the set of employees g that has been assigned j and that includes r (i.e., r must work in collaboration with the other employees assigned to j). The fourth component refers to the attributes of r (i.e., his or her experience level in relation to different tasks and domains, the kind of labor relation between r and the other employees of g, his or her educational level in relation to different knowledge areas, his or her level with respect to different skills, etc.). It is considered that the attributes of r could be quantified from available information about r (e.g., curriculum vitae of r, results of evaluations made to r, information about the participation of r in already executed projects, etc.).

The four components described above are considered the main factors that determine the effectiveness level of an employee. For this reason, it is assumed that the effectiveness of an employee depends on all the components of his or her work context. Then, for each employee, it is possible to consider different effectiveness levels in relation to different work contexts.

The effectiveness level of an employee r, in relation to a possible context $C_{r,j,k,g}$ for r, is notated as $e_{rCr,j,k,g}$. The term $e_{rCr,j,k,g}$ represents how well r can handle, within

activity j, the tasks associated to skill k, considering that r must work in collaboration with the other employees of set g. The mentioned term $e_{rCr,j,k,g}$ takes a real value over the range [0, 1]. The values of the terms $e_{rCr,j,k,g}$ inherent to each employee available for the project are known. It is considered that these values could be obtained from available information about the participation of the employees in already executed projects.

The problem of scheduling a project entails defining feasible start times (i.e., the precedence relations between the activities must not be violated) and feasible human resource assignments (i.e., the human resource requirements must be met) for project activities in such a way that the optimization objective is reached. In this sense, a priority objective is considered for project managers at the early stage of the project schedule design. The objective is that the most effective set of employees be assigned each project activity. This objective is modeled by Formulas (1) and (2).

Formula (1) maximizes the effectiveness of the sets of employees assigned to the N activities of a given project. In this formula, set S contains all the feasible schedules for the project in question. The term $e(s)$ represents the effectiveness level of the sets of employees assigned to project activities by schedule s. Then, $R(j,s)$ is the set of employees assigned to activity j by schedule s, and the term $e_{R(j,s)}$ represents the effectiveness level corresponding to $R(j,s)$.

Formula (2) estimates the effectiveness level of the set of employees $R(j,s)$. This effectiveness level is estimated calculating the mean effectiveness level of the employees belonging to $R(j,s)$.

For a more detailed discussion of Formulas (1) and (2), we refer to [6].

$$\max_{\forall s \in S} \left(e(s) = \sum_{j=1}^{N} e_{R(j,s)} \right) \tag{1}$$

$$e_{R(j,s)} = \frac{\sum_{r=1}^{|R(j,s)|} e_{rC_{r,j,k(r,j,s),R(j,s)}}}{|R(j,s)|} \tag{2}$$

3 Hybrid Evolutionary Algorithm with Self-Adaptive Processes

In order to solve the addressed problem, we present a hybrid evolutionary algorithm with self-adaptive processes. This algorithm integrates self-adaptive processes which adapt their behavior in accordance with the state of the evolutionary search. The integration of self-adaptive processes has the aim of enhancing the evolutionary search, in both exploitation and exploration [18–20].

The main behavior of the hybrid evolutionary algorithm is shown in Fig. 1 and is described below.

Taking into account a given project to be scheduled, the algorithm creates a random initial population of solutions. In this population, each solution encodes a feasible

```
BEGIN
    CREATE initial population;
    EVALUATE each solution of the population;
    REPEAT UNTIL ( number of iterations is reached ) DO
        SELECT parents;
        RECOMBINE pairs of parents to produce offspring;
        MUTATE the resulting offspring;
        EVALUATE new solutions;
        CREATE new population;
        IMPROVE solutions via Simulated Annealing;
    OD
    PROVIDE best solution;
END
```

Fig. 1. Main behavior of the hybrid evolutionary algorithm

project schedule. Then, the algorithm decodes and evaluates each solution of the population by a fitness function. Specifically, the schedule related to each solution is built and then evaluated with respect of the optimization objective of the problem. As was detailed in Sect. 2, the optimization objective implies maximizing the effectiveness of the sets of employees assigned to the project activities. In relation to this objective, the fitness function evaluates the employee assignments of each solution based on knowledge of the effectiveness of the employees considered in the solution.

After each solution of the population is evaluated, a parent selection process is utilized to determine which solutions of the population will integrate the mating pool. The solutions with the highest fitness values will have more chance of being selected. Once the mating pool is complete, the solutions in the mating pool are organized in pairs. After that, a crossover process is applied to each pair of solutions with a self-adaptive probability AP_c to generate new feasible ones. After that, a mutation process is applied to each solution obtained by the crossover process, with a self-adaptive probability AP_m. After that, a survival selection process is applied in order to determine which solutions from the solutions in the population and the solutions generated from the mating pool will integrate the new population. Finally, a self-adaptive simulated annealing algorithm is applied to the solutions of the new population. The above-described process is repeated until a predefined number of iterations is reached.

3.1 Representation of Solutions

We used the representation described in [6] in order to encode the solutions. By this representation, each solution is encoded by two lists with as many positions as activities in the project.

The first list is a traditional activity list. This list is a feasible precedence list of the activities in the project (i.e., each activity j can appear on this list in any position higher than the positions of all its predecessors). The activity list defines the order in which activities shall be added to the schedule.

The second list is an assigned resources list. This list details the employees assigned to each activity of the project (i.e., position j on this list details the employees of every skill k assigned to activity j).

In order to decode the schedule related to the representation, we used the serial schedule generation process described in [6]. By this process, each activity j is scheduled at the earliest possible time.

3.2 Fitness Function

To evaluate the encoded solutions, we used a fitness function specially designed. Given an encoded solution, this function decodes the schedule s related to the solution by the serial schedule generation process mentioned in Sect. 3.1. After that, the fitness function calculates the value of the term $e(s)$ corresponding to s (Formulas (1) and (2)). This value defines the fitness level of the solution. The term $e(s)$ takes a real value on $[0,\ldots, N]$.

In order to calculate the value of term $e(s)$, the fitness function utilizes the values of the terms $e_{rCr,j,k,g}$ inherent to s (Formula 2). In this respect, the values of the terms $e_{rCr,j,k,g}$ inherent to each available employee r are known, as was detailed in Sect. 2.

3.3 Parent Selection and Survival Selection

To develop the parent selection, we used the process named roulette wheel selection [18]. In this process, a selection probability is defined for each solution of the current population. The selection probability of each solution is proportional to its fitness value. Thus, the solutions with the best fitness values have more probability of being selected for the mating pool.

To develop the survival selection, we utilized the process named fitness-based steady-state selection [18]. By this process, the worst λ solutions of the current population are replaced by the best λ solutions generated from the mating pool. This process preserves the best solutions reached by the hybrid evolutionary algorithm [18].

3.4 Self-Adaptive Crossover and Self-Adaptive Mutation

In respect of the crossover process and the mutation process, we utilized self-adaptive processes feasible for the representation of the solutions.

The crossover process is composed by a crossover process feasible for activity lists and a crossover process feasible for assigned resources lists. In respect of the crossover for activity lists, we applied a process named two-point crossover [21]. For assigned resources lists, we applied a process named uniform crossover [18].

The mutation process is composed by a mutation process feasible for activity lists and a mutation process feasible for assigned resources lists. In relation to the mutation for activity lists, we applied a process named adjacent pairwise interchange [21]. For assigned resources lists, we applied a process named random resetting [18].

In order to apply the crossover process and the mutation process, we considered self-adaptive probabilities AP_c and AP_m, respectively. Specifically, we considered the known self-adaptive probabilities AP_c and AP_m described in [11]. These probabilities are defined by Formulas (3) and (4). In these formulas, 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 state of the evolutionary search. In Formula (3), f' is the higher fitness of the two solutions to be crossed, and AP_{cLA} and AP_{cUA} are predefined values for the crossover probability, considering $0 \leq AP_{cLA}, AP_{cUA} \leq 1$. In Formula (4), f'' is the fitness of the solution to be mutated, and AP_{mLA} and AP_{mUA} are predefined values for the mutation probability, considering $0 \leq AP_{mLA}, AP_{mUA} \leq 1$.

Probabilities AP_c and AP_m are adaptive according to the state of the evolutionary search. Specifically, when the evolutionary search starts to converge, AP_c and AP_m are increased in order to encourage the exploration of new regions of the search space and therefore to avoid the premature convergence of the evolutionary search. In contrast, when the evolutionary search is scattered in the search space, AP_c and AP_m are reduced in order to encourage the exploitation of known regions of the search space.

$$AP_c = \begin{cases} \frac{AP_{cUA}(f_{max} - f')}{(f_{max} - f_{avg})} & f' \geq f_{avg} \\ AP_{cLA} & f' < f_{avg} \end{cases} \tag{3}$$

$$AP_m = \begin{cases} \frac{AP_{mUA}(f_{max} - f'')}{(f_{max} - f_{avg})} & f'' \geq f_{avg} \\ AP_{mLA} & f'' < f_{avg} \end{cases} \tag{4}$$

3.5 Self-Adaptive Simulated Annealing Algorithm

We applied a self-adaptive simulated annealing algorithm to the solutions of the population obtained by the survival selection process, except to the solution with the highest fitness value of this population which is preserved. The applied self-adaptive simulated annealing algorithm is a variation of the simulated annealing algorithm presented in [8].

The main behavior of the self-adaptive simulated annealing algorithm is described as follows. Given an encoded solution s, the algorithm generates a new encoded solution s' from the solution s by using a move process, and then decides if the solution s must be replaced or not by the new solution s'. If the fitness value of the new solution s' is better than that of the solution s, the algorithm replaces to the solution s by the solution s'. In contrast, if the fitness value of the new solution s' is worse than or equal to that of the solution s, the algorithm replaces to the solution s by the solution s' based on an acceptance probability which is $exp(-delta / T_c)$. In this probability, term T_c is the current value of the temperature parameter and $delta$ is the difference between the fitness values of the solutions s and s'. Thus, the acceptance probability is proportional to the current value of the temperature parameter.

The above-described process is repeated until a predetermined number of iterations is reached. It is necessary to mention that, at the end of each iteration, the value of the temperature parameter is reduced by a predefined cooling factor.

In relation to the initial value T_i of the temperature parameter, we defined the value T_i based on the evolutionary search state reached after the survival selection process, considering that such state is measured by calculating the term $(f_{max} - f_{avg})$ on the population obtained by the survival selection process. Specifically, we calculated the value T_i by using the next formula: $T_i = 1 / (f_{max} - f_{avg})$. By this formula, when the evolutionary search is scattered in the search space, the value T_i is low, and thus the acceptance probability of the algorithm is also low. As consequence of this, the algorithm promotes the exploitation of known regions of the search space. When the evolutionary search starts to converge, the value T_i increases, and thus the acceptance probability of the algorithm also increases. As consequence of this, the algorithm promotes the exploration of new regions of the search space. Based on the mentioned, the algorithm is self-adaptive to promote either the exploitation or exploration of the search space, according to the state of the evolutionary search.

Move Process. The self-adaptive simulated annealing algorithm utilizes a move process to produce a new encoded solution from a given encoded solution. In this respect, we applied a move process feasible for the representation of the solutions. The move process is composed by a move process feasible for activity lists and a move process feasible for assigned resources lists. In respect of the move process for activity lists, we applied a move process named simple shift [21]. For assigned resources lists, we applied a move process which is considered as a variation of the process named random resetting [18].

4 Computational Experiments

4.1 Instance Sets

We used the six instance sets introduced in [7] in order to evaluate the performance of the hybrid evolutionary algorithm. Each one of these six instance sets contains 40 instances. Each instance contains a number of activities to be scheduled as well as a number of available employees for these activities. The main characteristics of these six instance sets are presented in Table 1. For a more detailed description of these six instance sets, we refer to [7].

It is necessary to mention that each instance of these six instance sets has a known optimal solution with a fitness level equal to N. Note that N is the number of activities to be scheduled in the instance. These known optimal solutions of the instances are considered as references to evaluate the performance of the algorithm.

4.2 Main Results

We evaluated the performance of the hybrid evolutionary algorithm on each of the six instance sets. Specifically, we run the algorithm a predetermined number t of times (i.e.,

Table 1. Main characteristics of the instance sets

Instance set	Number of activities per instance	Number of possible sets of employees per activity
j30_5	30	1 to 5
j30_10	30	1 to 10
j60_5	60	1 to 5
j60_10	60	1 to 10
j120_5	120	1 to 5
j120_10	120	1 to 10

t = 30 times) on each instance of the six instance sets. In order to develop these runs, we set the algorithm parameters as follows: population size = 90; number of generations = 300; crossover process: AP_{cLA} = 0.9 and AP_{cUA} = 0.6; mutation process: AP_{mLA} = 0.1 and AP_{mUA} = 0.05; survival selection process: $\lambda = 45$; simulated annealing algorithm: number of iterations = 25 and cooling factor = 0.9. It is necessary to mention that we set the algorithm parameters with these values based on exhaustive preliminary experiments. By these preliminary experiments, we considered many different settings for the algorithm parameters, and then we selected the best of these settings for the algorithm parameters.

We analyzed the results obtained by the hybrid evolutionary algorithm for each of the six instance sets. Specifically, for each instance set, we analyzed the average percentage deviation from the optimal value (Av. Dev. (%)) as well as the percentage of instances for which the optimal value is reached at least once among the t runs developed (Opt. (%)).

For j30_5, j30_10, j60_5 and j60_10, the algorithm obtained an Av. Dev (%) equal to 0 % and an Opt. (%) equal to 100 %. These results indicate that the algorithm reached an optimal solution in each run developed on each instance of these sets.

For j120_5 and j120_10, the algorithm obtained Av. Dev (%) values equal to 0.1 % and 0.36 %, respectively. Because the optimal solutions of the instances of both sets have a fitness level equal to 120, these results indicate that the average fitness level of the solutions obtained by the algorithm for j120_5 and j120_10 is 119.88 and 119.57, respectively. Therefore, the algorithm obtained very near-optimal solutions for the instances of both sets.

Moreover, for j120_5 and j120_10, the algorithm obtained an Opt. (%) value equal to 100 %. These results indicate that, for each instance of these two sets, the algorithm reached an optimal solution at least once among the t runs developed on the instance.

4.3 Comparison

In this section, we compare the performance of the hybrid evolutionary algorithm with that of the best algorithm previously presented in the literature for solving 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 integrates an adaptive simulated annealing algorithm into 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 state of the evolutionary search.

In [8], the algorithm H has been evaluated on the six instance sets presented in Table 1 and has obtained the following results. For j30_5, j30_10, j60_5 and j60_10, the algorithm H obtained an Av. Dev (%) equal to 0 % and an Opt. (%) equal to 100 %. For j120_5 and j120_10, the algorithm H obtained Av. Dev (%) values equal to 0.64 % and 0.8 %, respectively. Moreover, for j120_5 and j120_10, the algorithm obtained an Opt. (%) value equal to 100 %.

Comparing the results obtained by the algorithm H and the hybrid evolutionary algorithm presented here, we can mention the following points. Both algorithms have obtained an optimal effectiveness level for j30_5, j30_10, j60_5 and j60_10 (i.e., the less complex instance sets). However, the effectiveness level obtained by the hybrid evolutionary algorithm for j120_5 and j120_10 (i.e., the more complex instance sets) is significantly higher than that obtained by the algorithm H. Therefore, the hybrid evolutionary algorithm outperforms the algorithm H on the more complex instance sets. This is mainly because of the following reasons.

The hybrid evolutionary algorithm integrates self-adaptive crossover and mutation processes. These processes adapt their behaviour according to the state of the evolutionary search, in order to promote either the exploitation or exploration of the search space and thus enhance the evolutionary search. In contrast with the hybrid evolutionary algorithm, the algorithm H utilizes non-adaptive crossover and mutation processes. These processes do not consider the state of the evolutionary search and thus do not have the possibility of enhancing the evolutionary search.

5 Related Works

A wide variety of reported project scheduling problems consider the effectiveness of human resources. Nonetheless, these project scheduling problems differ significantly in the assumptions considered regarding the effectiveness of human resources. In this respect, only few of these project scheduling problems consider that human resources usually have very different levels of effectiveness [3–6, 10], a really essential aspect in the context of real project scheduling problems. In this section, we review the assumptions considered about the effectiveness of human resources in reported project scheduling problems.

In the multi-skill project scheduling problems reported in [12–17], each project activity requires a given number of skills and a given number of human resources for each required skill. Each available human resource masters one or several skills, and all the human resources that master a given skill have the same effectiveness level in relation to such skill.

In the multi-skill project scheduling problem reported in [3], hierarchical levels of skills are considered. Given a skill, for each human resource that masters the skill, an effectiveness level is defined in relation to the skill. Thus, the human resources that master a given skill have different levels of effectiveness in relation to the skill. Then, each project

activity requires one or several skills, a minimum effectiveness level for each skill, and a number of human resources for each pair skill-level. This problem assumes that all sets of human resources that can be assigned to a given activity have the same effectiveness on the development of the activity. Specifically, with respect to effectiveness, such sets are merely treated as unary resources with homogeneous levels of effectiveness.

In the multi-skill project scheduling problems reported in [4, 5], most activities require only one human resource with a particular skill, and each available human resource masters different skills. The human resources that master a given skill have different levels of effectiveness with respect of such skill. Then, the effectiveness of a human resource in a given activity is defined by considering only the effectiveness level of the human resource in relation to the skill required for the activity.

In contrast to the problems above-mentioned, in the project scheduling problem reported in [6], it is supposed that the effectiveness level of a human resource depends on different factors inherent to its work context. Therefore, for each human resource, it is possible to define different effectiveness levels with respect to different work contexts. This assumption about the effectiveness levels of human resources is really valuable. This is mainly because, in the context of real project scheduling problems, human resources have different levels of effectiveness with regards to different work contexts, and therefore, the effectiveness level of a human resource is usually considered in relation to its work context [1, 2]. Based on the above-mentioned, the project scheduling problem reported in [6] supposes valuable assumptions about the effectiveness levels of human resources in the context of project scheduling problems.

6 Conclusions

In this paper, we addressed the project scheduling problem described in [6]. This problem considers really valuable assumptions about the effectiveness of human resources. Moreover, this problem considers an optimization objective priority for project managers: maximizing the effectiveness levels of the sets of human resources assigned to the project activities.

We presented a hybrid evolutionary algorithm with self-adaptive processes for solving the addressed problem. This algorithm integrates self-adaptive processes which adapt their behavior in accordance with the state of the evolutionary search. The integration of self-adaptive processes has the aim of enhancing the evolutionary search, in both exploration and exploitation.

We evaluated the performance of the hybrid evolutionary algorithm on different instance sets. Then, we compared the performance of the hybrid evolutionary algorithm with that of the best algorithm previously reported in the literature for solving the addressed problem. Based on the obtained results, we may state that the hybrid evolutionary algorithm considerably outperforms the previous algorithm.

In future works, we will evaluate the integration of other self-adaptive process into the framework of the evolutionary algorithm. In particular, we will evaluate other self-adaptive local search and optimization techniques, other self-adaptive crossover processes, as well as other self-adaptive mutation processes.

References

1. Heerkens, G.R.: Project Management. McGraw-Hill, New York (2002)
2. Wysocki, R.K.: Effective Project Management, 3rd edn. Wiley, Hoboken (2003)
3. Bellenguez, O., Néron, E.: Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In: Burke, E.K., Trick, M.A. (eds.) PATAT 2004. LNCS, vol. 3616, pp. 229–243. Springer, Heidelberg (2005)
4. Hanne, T., Nickel, S.: A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. Eur. J. Oper. Res. **167**, 663–678 (2005)
5. Gutjahr, W.J., Katzensteiner, S., Reiter, P., Stummer, Ch., Denk, M.: Competence-driven project portfolio selection, scheduling and staff assignment. Central Eur. J. Oper. Res. **16**(3), 281–306 (2008)
6. Yannibelli, V., Amandi, A.: A knowledge-based evolutionary assistant to software development project scheduling. Expert Syst. Appl. **38**(7), 8403–8413 (2011)
7. Yannibelli, V., Amandi, A.: A memetic approach to project scheduling that maximizes the effectiveness of the human resources assigned to project activities. In: Corchado, E., Snášel, V., Abraham, A., Woźniak, M., Graña, M., Cho, S.-B. (eds.) HAIS 2012, Part I. LNCS, vol. 7208, pp. 159–173. Springer, Heidelberg (2012)
8. Yannibelli, V., Amandi, A.: A diversity-adaptive hybrid evolutionary algorithm to solve a project scheduling problem. In: Corchado, E., Lozano, J.A., Quintián, H., Yin, H. (eds.) IDEAL 2014. LNCS, vol. 8669, pp. 412–423. Springer, Heidelberg (2014)
9. Blazewicz, J., Lenstra, J., Rinnooy Kan, A.: Scheduling subject to resource constraints: classification and complexity. Discrete Appl. Math. **5**, 11–24 (1983)
10. Yannibelli, V., Amandi, A.: Project scheduling: a multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan. Eng. Optim. **45**(1), 45–65 (2013)
11. Srinivas, M., Patnaik, L.M.: Adaptive probabilities of crossover and mutation in genetic algorithms. IEEE Trans. Syst. Man Cybern. **24**(4), 656–667 (1994)
12. Bellenguez, O., Néron, E.: A branch-and-bound method for solving multi-skill project scheduling problem. RAIRO – Oper. Res. **41**(2), 155–170 (2007)
13. Drezet, L.E., Billaut, J.C.: A project scheduling problem with labour constraints and time-dependent activities requirements. Int. J. Prod. Econ. **112**, 217–225 (2008)
14. Li, H., Womer, K.: Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. J. Sched. **12**, 281–298 (2009)
15. Valls, V., Pérez, A., Quintanilla, S.: Skilled workforce scheduling in service centers. Eur. J. Oper. Res. **193**(3), 791–804 (2009)
16. Aickelin, U., Burke, E., Li, J.: An evolutionary squeaky wheel optimization approach to personnel scheduling. IEEE Trans. Evol. Comput. **13**(2), 433–443 (2009)
17. Heimerl, C., Kolisch, R.: Scheduling and staffing multiple projects with a multi-skilled workforce. OR Spectrum **32**(4), 343–368 (2010)
18. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing, 2nd edn. Springer, Berlin (2015)
19. 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)
20. Talbi, E.: Hybrid metaheuristics. SCI, vol. 434. Springer, Berlin (2013)
21. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur. J. Oper. Res. **174**, 23–37 (2006)