A Diversity-Adaptive Hybrid Evolutionary Algorithm to Solve a Project Scheduling Problem

Virginia Yannibelli[1,2] and Analía Amandi[1,2]

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

Abstract. In this paper, we address a project scheduling problem. This problem considers a priority optimization objective for project managers. This objective implies assigning the most effective set of human resources to each project activity. To solve the problem, we propose a hybrid evolutionary algorithm. This algorithm incorporates a diversity-adaptive simulated annealing algorithm into the framework of an evolutionary algorithm with the aim of improving the performance of the evolutionary search. The simulated annealing algorithm adapts its behavior according to the fluctuation of diversity of evolutionary algorithm population. The performance of the hybrid evolutionary algorithm on six different instance sets is compared with those of the algorithms previously proposed in the literature for solving the addressed problem. The obtained results show that the hybrid evolutionary algorithm significantly outperforms the previous algorithms.

Keywords: project scheduling, human resource assignment, multi-skilled resources, hybrid evolutionary algorithms, evolutionary algorithms, simulated annealing algorithms.

1 Introduction

A project scheduling problem implies defining feasible start times and feasible human resource assignments for project activities in such a way that the optimization objective, defined as part of the problem, is reached. Besides, to define human resource assignments, it is necessary to have knowledge about the effectiveness of the available human resources in relation to different project activities. This is mainly because the development and the results of an activity depend on the effectiveness of the human resources assigned to it [1, 2].

Until now, many different kinds of project scheduling problems have been formally described and addressed in the literature. However, to the best of our knowledge, only few project scheduling problems have considered human resources with different levels of effectiveness [3, 4, 5, 6, 7, 10], a fundamental aspect in real project scheduling problems. These problems state different assumptions about the effectiveness of the human resources.

E. Corchado et al. (Eds.): IDEAL 2014, LNCS 8669, pp. 412–423, 2014.

The project scheduling problem described in [6, 7] considers that the effectiveness of a human resource depends on various factors inherent to its work context (i.e., the activity to which the resource is assigned, the skill to which the resource is assigned within the activity, the set of human resources that has been assigned to the activity, and the attributes of the resource). This is a really significant aspect of the project scheduling problem described in [6, 7]. This is mainly because, in real project scheduling problems, the human resources usually have different effectiveness levels in relation to different work contexts [8, 1, 2] and, therefore, the effectiveness of a human resource needs to be considered in relation to its work context. To the best of our knowledge, the influence of the work context on the effectiveness of the human resources has not been considered in other project scheduling problems. Because of this, we consider that the project scheduling problem described in [6, 7] states valuable and novel assumptions about the effectiveness of the human resources in the context of project scheduling problems. In addition, this problem considers a priority optimization objective for managers at the early stage of scheduling. This objective involves assigning the most effective set of human resources to each project activity.

The project scheduling problem described in [6, 7] can be seen as a special case of the RCPSP (Resource Constrained Project Scheduling Problem) [9] and, therefore, is a NP-Hard problem. For this reason, exhaustive search algorithms only can solve very small instances of the problem in a reasonable period of time. Thus, heuristic search algorithms have been proposed in the literature to solve the problem: an evolutionary algorithm was proposed in [6], and a memetic algorithm was proposed in [7] that incorporates a hill-climbing algorithm into the framework of an evolutionary algorithm. The memetic algorithm is the best of both algorithms, as reported in [7].

In this paper, we address the project scheduling problem described in [6, 7] with the aim of proposing a better heuristic search algorithm to solve it. In this respect, we propose a hybrid evolutionary algorithm. This algorithm integrates a diversity-adaptive simulated annealing algorithm within the framework of an evolutionary algorithm. The behavior of the simulated annealing algorithm is adaptive based on the fluctuation of diversity of evolutionary algorithm population. The integration of a diversity-adaptive simulated annealing algorithm is meant to improve the performance of the evolutionary search [18, 19].

We propose the above-mentioned hybrid evolutionary algorithm based on the following reasons. The hybridization of evolutionary algorithms with other search and optimization techniques has been proven to be more effective than the classical evolutionary algorithms in the resolution of a wide variety of NP-Hard problems [18, 19, 20, 21] and, in particular, in the resolution of scheduling problems [22, 7, 21, 20, 19]. Besides, the hybridization of evolutionary algorithms with simulated annealing algorithms has been shown to be more effective than the hybridization of evolutionary algorithms with hill-climbing algorithms in the resolution of different NP-Hard problems [18, 19]. Thus, we consider that the proposed hybrid evolutionary algorithm could outperform the heuristic algorithms previously proposed to solve the problem.

The remainder of the paper is organized as follows. In Section 2, we give a brief review of published works that consider the effectiveness of human resources in the context of project scheduling problems. In Section 3, we describe the problem

addressed in this paper. In Section 4, we present the proposed hybrid algorithm. In Section 5, we present the computational experiments carried out to evaluate the performance of the hybrid algorithm and an analysis of the results obtained. Finally, in Section 6 we present the conclusions of the present work.

2 Related Works

Different project scheduling problems described in the literature have considered the effectiveness of human resources. However, these project scheduling problems state different assumptions about the effectiveness of human resources. In this respect, only few project scheduling problems have considered human resources with different levels of effectiveness [3, 4, 5, 6, 7, 10], a fundamental aspect in real project scheduling problems. In this section, we focus the attention on analyzing the way in which the effectiveness of human resources is considered in project scheduling problems reported in the literature.

In [12, 11, 13, 14, 15, 16, 17], the authors describe multi-skill project scheduling problems. In these problems, each project activity requires specific skills and a given number of human resources (employees) for each required skill. Each available employee masters one or several skills, and all the employees that master a given skill have the same effectiveness level in relation to the skill (homogeneous levels of effectiveness in relation to each skill).

In [3], the authors describe a multi-skill project scheduling problem with hierarchical levels of skills. In this problem, given a skill, for each employee that masters the skill, an effectiveness level is defined in relation to the skill. Thus, the employees 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 resources for each pair skill-level. This work considers that all sets of employees 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 [4, 5], the authors describe multi-skill project scheduling problems. In these problems, most activities require only one employee with a particular skill, and each available employee masters different skills. Besides, the employees that master a given skill have different levels of effectiveness in relation to the skill. Then, the effectiveness of an employee in a given activity is defined by considering only the effectiveness level of the employee in relation to the skill required for the activity.

Unlike the above-mentioned problems, the project scheduling problem described in [6, 7] considers that the effectiveness of a human resource depends on various factors inherent to its work context. Then, for each human resource, it is possible to define different effectiveness levels in relation to different work contexts. This is a very important aspect of the project scheduling problem described in [6, 7]. This is because, in real project scheduling problems, the human resources have different effectiveness levels in relation to different work contexts [8, 1, 2] and, therefore, the

effectiveness of a human resource needs to be considered in relation to its work context. Based on the above-mentioned, we consider that the project scheduling problem described in [6, 7] states valuable assumptions about the effectiveness of the human resources in the context of project scheduling problems.

3 Problem Description

In this paper, we address the project scheduling problem presented in [6, 7]. A description of this problem is presented below.

Suppose that a project contains a set A of N activities, $A = \{1, \ldots, N\}$, that has 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].

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

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

4 Hybrid Evolutionary Algorithm

To solve the problem, we propose a hybrid evolutionary algorithm. This algorithm incorporates a diversity-adaptive simulated annealing stage into the framework of an evolutionary algorithm. The behavior of the simulated annealing stage is adaptive based on the diversity within the underlying evolutionary algorithm population. The incorporation of a diversity-adaptive simulated annealing stage pursues two aims. When the evolutionary algorithm population is diverse, the simulated annealing stage behaves like an exploitation process to fine-tune the solutions in the population. When the evolutionary algorithm population starts to converge, the simulated annealing stage changes its behavior from exploitation to exploration in order to introduce diversity in the population and thus to prevent the premature convergence of the evolutionary search. Thus, the evolutionary search is augmented by the addition of one stage of diversity-adaptive simulated annealing [18, 19].

The general behavior of the hybrid evolutionary algorithm is described as follows. Given a project to be scheduled, the algorithm starts the evolution from a random initial population of feasible solutions. Each of these solutions codifies a feasible project schedule. Then, each solution of the population is decoded (i.e., the related schedule is built), and evaluated according to the optimization objective of the problem by a fitness function. As explained in Section 3, the objective is to maximize the effectiveness of the sets of employees assigned to project activities. In relation to this objective, the fitness function evaluates the assignments of each solution based on knowledge about the effectiveness of the employees involved in the solution.

After the solutions of the population are evaluated, a parent selection process is used to determine which solutions of the population will compose the mating pool. The solutions with the greatest fitness values will have more chances of being selected. Once the mating pool is composed, the solutions in the mating pool are paired, and a crossover process is applied to each pair of solutions with a probability P_c to generate new feasible ones. Then, a mutation process is applied to each solution generated by the crossover process, with a probability P_m. The mutation process is applied with the aim of introducing diversity in the new solutions generated by the crossover process. Then, a survival selection strategy is used to determine which solutions from the solutions in the population and the solutions generated from the mating pool will compose the new population. Finally, a diversity-adaptive simulated annealing algorithm is applied to the solutions of the new population. The simulated annealing algorithm behaves like either an exploitation process or an exploration process depending on the diversity of the population. Thus, the simulated annealing algorithm modifies the solutions of the population.

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

4.1 Encoding of Solutions and Fitness Function

In relation to the encoding of solutions, we used a representation proposed in [6]. Each solution is represented by two lists having as many positions as activities in the project. The first list is a standard activity list. This list is a feasible precedence list of

the activities involved in the project (i.e., each activity *j* can appear on the list in any position higher than the positions of all its predecessors). The activity list describes the order in which activities shall be added to the schedule.

The second list is an assigned resources list. This list contains information about the employees assigned to each activity of the project. Specifically, position *j* on this list details the employees of every skill *k* assigned to activity *j*.

In order to build the schedule related to the representation, we used the serial schedule generation method proposed in [6]. In this method, each activity *j* is scheduled at the earliest possible time.

To evaluate a given encoded solution, a fitness function is used. This function decodes the schedule *s* related to the solution by using the serial method above-mentioned. Then, the function calculates the value of the term *e(s)* corresponding to *s* (Formulas (1) and (2)). This value determines the fitness level of the solution. The term *e(s)* takes a real value over [0, …, *N*].

To calculate the term *e(s)*, the function utilizes the values of the terms $e_{rCr,j,k,g}$ inherent to *s* (Formula 2). As was mentioned in Section 3, the values of the terms $e_{rCr,j,k,g}$ inherent to each available employee *r* are known.

4.2 Parent Selection, Crossover, Mutation and Survival Selection

To develop the parent selection, we used the process called deterministic tournament selection with replacement [18], where the parameter *t* defines the tournament size. This process is a variant of the traditional tournament selection process [18].

To develop the crossover and the mutation, we considered feasible processes for the representation used for the solutions. Thus, the crossover operator contains a feasible crossover operation for activity lists and a feasible crossover operation for assigned resources lists. For activity lists, we considered the one-point crossover proposed by Hartmann [22]. For assigned resources lists, we considered the traditional uniform crossover [18]. The crossover operator is applied with a probability of P_c.

The mutation operator contains a feasible mutation operation for activity lists and a feasible mutation operation for assigned resources lists. For activity lists, we considered the simple shift operator described in [22]. For assigned resources lists, we considered the traditional random resetting [18].

To develop the survival selection, we applied the process called deterministic crowding [18]. This process preserves the best solutions found by the hybrid evolutionary algorithm and preserves the diversity of the population. For a detailed description of the deterministic crowding process, we refer to [18].

4.3 Diversity-Adaptive Simulated Annealing Algorithm

A diversity-adaptive simulated annealing algorithm is applied to each solution of the population obtained by the survival selection process, except to the best solution of this population. The best solution of the population is maintained into this population.

The simulated annealing algorithm applied here is an adaptation of the one proposed in [23], and is described as follows.

The simulated annealing algorithm is an iterative process that starts considering a given encoded solution *s* for the problem and a given initial value T_0 for a parameter called temperature. In each iteration, a new solution *s'* is generated from the current solution *s* by a move operator. If the new solution *s'* is better than the current solution *s* (i.e., the fitness value of *s'* is higher than the fitness value of *s*), the current solution *s* is replaced by *s'*. Otherwise, if the new solution *s'* is worse than the current solution *s*, the current solution *s* is replaced by *s'* with a probability equal to *exp*(*-delta / T*), where *T* is the current temperature value and *delta* is the difference between the fitness value of *s* and the fitness value of *s'*. Thus, the probability of accepting a new solution that is worse than the current solution mainly depends on the temperature value. If the temperature is high, the acceptance probability is also high, and vice versa. The temperature value is decreased by a cooling factor at the end of each iteration. The described process is repeated until a predetermined number of iterations is reached.

Before applying the simulated annealing algorithm to the solutions of the population, the initial temperature T_0 is defined. In this case, the initial temperature T_0 is inversely proportional to the diversity of the population, and this diversity is represented by the spread of fitnesses within the population. Specifically, T_0 is calculated as detailed in Formula (3), where f_{max} is the maximal fitness into the population and f_{min} is the minimal fitness into the population. Therefore, when the population is very diverse, the value of T_0 is very low, and thus, the simulated annealing algorithm only accepts movements that improve the solutions to which it is applied, behaving like an exploitation process. When the population converges, the value of T_0 rises, and thus, the simulated annealing algorithm increases the probability of accepting worsening movements. A consequence of this is that the simulated annealing algorithm will try to move away from the solutions to which it is applied, exploring the search space. Eventually, the diversity of the population will increase, and thus, the temperature T_0 will decrease. Based on the above-mentioned, the self-adaptation of the simulated annealing algorithm to either an exploitation or exploration behavior is governed by the diversity of the population.

$$T_0 = \frac{1}{\left| f_{max} - f_{min} \right|} \tag{3}$$

In relation to the move operator used by the simulated annealing to generate a new solution from the current solution, we considered a feasible move operator for the representation used for the solutions. Thus, the move operator contains a feasible move operation for activity lists and a feasible move operation for assigned resources lists. For activity lists, we considered a move operator called adjacent pairwise interchange [22]. For assigned resources lists, we considered a move operator that is a variant of the traditional random resetting [18]. In this variant, only one randomly selected position of the list is modified.

5 Computational Experiments

To develop the computational experiments, we used the six instance sets presented in [7]. Table 1 shows the main characteristics of these instance sets. Each instance of these six instance sets contains information about a number of activities to be planned and information about a number of employees available to develop the activities. For a detailed description of these instance sets, we refer to [7].

Each instance of the six instance sets has a known optimal solution with a fitness level *e(s)* equal to *N* (*N* is the number of activities of the instance). These optimal solutions are considered here as references.

The hybrid evolutionary algorithm was run 20 times on each of the instances of the six instance sets. After each one of the 20 runs, the algorithm provided the best solution of the last population. To perform these runs, the algorithm parameters were set with the values shown in Table 2. The parameters were fixed thanks to preliminary experiments that showed that these values led to the best and most stable results.

Table 1. Characteristics of instance sets

Instance set	Activities per instance	Possible sets of employees per activity	Instances
j30_5	30	1 to 5	40
j30_10	30	1 to 10	40
j60_5	60	1 to 5	40
j60_10	60	1 to 10	40
j120_5	120	1 to 5	40
j120_10	120	1 to 10	40

Table 2. Parameter values of the hybrid evolutionary algorithm

Parameter	Value
Population size	90
Number of generations	300
Parent Selection process	
t (tournament size)	5
Crossover process	
Crossover probability P_c	0.8
Mutation process	
Mutation Probability P_m	0.05
Simulated annealing algorithm	
Number of iterations	25
Cooling factor	0.9

Table 3 reports the results obtained by the experiments. The second column reports the average percentage deviation from the optimal solution (Av. Dev. (%)) for each instance set. The third column reports the percentage of instances for which the value of the optimal solution is achieved at least once among the 20 generated solutions (Optimal (%)).

Table 3. Results obtained by the computational experiments

Instance set	Av. Dev. (%)	Optimal (%)
j30_5	0	100
j30_10	0	100
j60_5	0	100
j60_10	0	100
j120_5	0.64	100
j120_10	0.8	100

The results obtained by the algorithm for j30_5, j30_10, j60_5 and j60_10 indicate that the algorithm has reached an optimal solution in each of the 20 runs carried out on each instance of the sets.

The Av. Dev. (%) obtained by the algorithm for j120_5 and j120_10 is greater than 0%. Taking into account that the instances of j120_5 and j120_10 have known optimal solutions with a fitness level *e(s)* equal to 120, we analyzed the meaning of the average deviation obtained for each one of these sets. In the case of j120_5 and j120_10, average deviations equal to 0.64% and 0.8% indicate that the average value of the solutions reached by the algorithm is 119.232 and 119.04 respectively. Therefore, we may state that the algorithm has obtained very high quality solutions for the instances of j120_5 and j120_10.

In addition, the Optimal (%) obtained by the algorithm for j120_5 and j120_10 is 100%. These results indicate that the algorithm has reached an optimal solution in at least one of the 20 runs carried out on each instance of the sets.

5.1 Comparison with a Competing Algorithm

To the best of our knowledge, only two algorithms have been previously proposed for solving the addressed problem: a classical evolutionary algorithm [6], and a classical memetic algorithm [7] that incorporates a hill-climbing algorithm into the framework of an evolutionary algorithm. According to the experiments reported in [7], both algorithms have been evaluated on the six instance sets presented in Table 1 and have obtained the results that are shown in Table 4. Based on the results in Table 4, the memetic algorithm is the best of both algorithms. Below, we compare the performance of the memetic algorithm with that of the hybrid evolutionary algorithm.

The results in Table 3 and Table 4 indicate that the hybrid evolutionary algorithm and the memetic algorithm have reached the same effectiveness level (i.e., an optimal effectiveness level) on the first three instance sets (i.e., the less complex sets). However, the effectiveness level reached by the hybrid evolutionary algorithm on the last three instance sets (i.e., the more complex sets) is higher than the effectiveness level reached by the memetic algorithm on these sets. Thus, the performance of the hybrid evolutionary algorithm on the three more complex instance sets is better than that of the memetic algorithm. The main reason for this is that the simulated annealing algorithm in the hybrid evolutionary algorithm adapts its behavior according to the fluctuation of population diversity and thus prevents the premature convergence of the evolutionary search, whereas the hill-climbing algorithm in the memetic algorithm

usually leads to a premature convergence of the evolutionary search. Thus, the hybrid evolutionary algorithm can reach better solutions than the memetic algorithm on the more complex instance sets.

Table 4. Results obtained by the algorithms previously proposed for the addressed problem

Instance set	Evolutionary algorithm [6]		Memetic algorithm [7]	
	Av. Dev. (%)	Optimal (%)	Av. Dev. (%)	Optimal (%)
j30_5	0	100	0	100
j30_10	0	100	0	100
j60_5	0.42	100	0	100
j60_10	0.59	100	0.1	100
j120_5	1.1	100	0.75	100
j120_10	1.29	100	0.91	100

6 Conclusions

We proposed a hybrid evolutionary algorithm to solve the project scheduling problem described in [6, 7]. This algorithm incorporates a diversity-adaptive simulated annealing algorithm into the framework of an evolutionary algorithm to improve the performance of the evolutionary search. The behavior of the simulated annealing algorithm is adaptive to either an exploitation or exploration behavior based on the fluctuation of population diversity. The presented computational experiments show that the hybrid evolutionary algorithm significantly outperforms the algorithms previously proposed for solving the addressed problem.

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

References

1. Heerkens, G.R.: Project Management. McGraw-Hill (2002)
2. Wysocki, R.K.: Effective Project Management, 3rd edn. Wiley Publishing (2003)
3. Bellenguez, O., Néron, E.: Lower Bounds for the Multi-skill Project Scheduling Problem with Hierarchical Levels of Skills. In: Burke, E., Trick, M. (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. European Journal of Operational Research 167, 663–678 (2005)
5. Gutjahr, W.J., Katzensteiner, S., Reiter, P., Stummer, C., Denk, M.: Competence-driven project portfolio selection, scheduling and staff assignment. Central European Journal of Operations Research 16(3), 281–306 (2008)
6. Yannibelli, V., Amandi, A.: A knowledge-based evolutionary assistant to software development project scheduling. Expert Systems with Applications 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. Barrick, M.R., Stewart, G.L., Neubert, M.J., Mount, M.K.: Relating member ability and personality to work-team processes and team effectiveness. Journal of Applied Psychology 83, 377–391 (1998)
9. Blazewicz, J., Lenstra, J., Rinnooy Kan, A.: Scheduling Subject to Resource Constraints: Classification and Complexity. Discrete Applied Mathematics 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. Engineering Optimization 45(1), 45–65 (2013)
11. Bellenguez, O.: A reactive approach for the multi-skill Project Scheduling Problem. In: 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), pp. 1–4. Université de Montréal, Montréal (2008)
12. Bellenguez, O., Néron, E.: A branch-and-bound method for solving multi-skill project scheduling problem. RAIRO - Operations Research 41(2), 155–170 (2007)
13. Drezet, L.E., Billaut, J.C.: A project scheduling problem with labour constraints and time-dependent activities requirements. International Journal of Production Economics 112, 217–225 (2008)
14. Li, H., Womer, K.: Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. Journal of Scheduling 12, 281–298 (2009)
15. Valls, V., Pérez, A., Quintanilla, S.: Skilled workforce scheduling in service centers. European Journal of Operational Research 193(3), 791–804 (2009)
16. Aickelin, U., Burke, E., Li, J.: An Evolutionary Squeaky Wheel Optimization Approach to Personnel Scheduling. IEEE Transactions on Evolutionary Computation 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 (2007)
19. Rodriguez, F.J., García-Martínez, C., Lozano, M.: Hybrid Metaheuristics Based on Evolutionary Algorithms and Simulated Annealing: Taxonomy, Comparison, and Synergy Test. IEEE Transactions on Evolutionary Computation 16(6), 787–800 (2012)
20. Talbi, E.-G. (ed.): Hybrid Metaheuristics. SCI, vol. 434. Springer, Heidelberg (2013)
21. Blum, C., Puchinger, J., Raidl, G.R., Roli, A.: Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11(6), 4135–4151 (2011)
22. Kolisch, R., Hartmann, S.: Experimental Investigation of Heuristics for Resource-Constrained Project Scheduling: An Update. European Journal of Operational Research 174, 23–37 (2006)
23. Yannibelli, V., Amandi, A.: Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem. Expert Systems with Applications 40(7), 2421–2434 (2013)