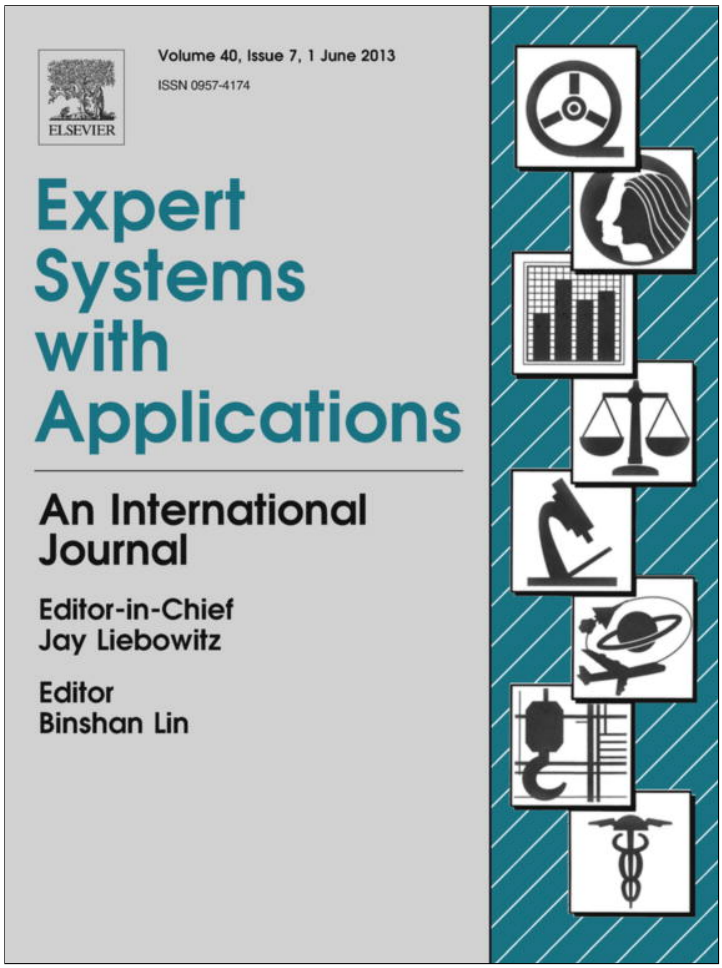

Expert Systems with Applications 40 (2013) 2421–2434

Contents lists available at SciVerse ScienceDirect

Expert Systems with Applications

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

Hybridizing a multi-objective simulated annealing algorithm with a multi-objective evolutionary algorithm to solve a multi-objective project scheduling problem

Virginia Yannibelli [a,b,*], Analía Amandi [a,b]

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

ARTICLE INFO

Keywords:
Multi-objective project scheduling
Multi-objective hybrid algorithm
Multi-objective simulated annealing algorithm
Multi-objective evolutionary algorithm
Non-dominated solutions

ABSTRACT

In this paper, a multi-objective project scheduling problem is addressed. This problem considers two conflicting, priority optimization objectives for project managers. One of these objectives is to minimize the project makespan. The other objective is to assign the most effective set of human resources to each project activity. To solve the problem, a multi-objective hybrid search and optimization algorithm is proposed. This algorithm is composed by a multi-objective simulated annealing algorithm and a multi-objective evolutionary algorithm. The multi-objective simulated annealing algorithm is integrated into the multi-objective evolutionary algorithm to improve the performance of the evolutionary-based search. To achieve this, the behavior of the multi-objective simulated annealing algorithm is self-adaptive to either an exploitation process or an exploration process depending on the state of the evolutionary-based search. The multi-objective hybrid algorithm generates a number of near non-dominated solutions so as to provide solutions with different trade-offs between the optimization objectives to project managers. The performance of the multi-objective hybrid algorithm is evaluated on nine different instance sets, and is compared with that of the only multi-objective algorithm previously proposed in the literature for solving the addressed problem. The performance comparison shows that the multi-objective hybrid algorithm significantly outperforms the previous multi-objective algorithm.

1. Introduction

A multi-objective project scheduling problem involves defining feasible start times and feasible human resource assignments for project activities so that the different optimization objectives, defined as part of the problem, are reached. Moreover, 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 because the development and the results of an activity depend on the effectiveness of the resources assigned to it (Heerkens, 2002; Wysocki, 2003).

In the literature, many different kinds of multi-objective project scheduling problems have been formally described and addressed until now. However, to the best of the authors' knowledge, only few multi-objective project scheduling problems have considered human resources with different levels of effectiveness (Bellenguez & Néron, 2005; Gutjahr, Katzensteiner, Reiter, Stummer, & Denk, 2008; Hanne & Nickel, 2005; Yannibelli & Amandi, 2012a), a central aspect in real multi-objective project scheduling contexts. These problems state different assumptions about the effectiveness of the human resources.

The multi-objective project scheduling problems described in Bellenguez and Néron (2005), Gutjahr et al. (2008), Hanne and Nickel (2005) assume that each human resource only has one or several skills, and an effectiveness level in relation to each skill. Then, the effectiveness of a human resource in a given activity is determined only on the basis of the effectiveness level of the resource in relation to one of the skills required for that activity. Thus, only the skills of a human resource are considered as determining factors of their effectiveness. However, other contextual factors that also determine the effectiveness of a human resource in a given activity are not considered in the mentioned problems. Such factors involve the attributes of the activity to which the resource is assigned, the other resources with whom the resource in question must work, as well as the experiences and attributes of the resource (Barrick, Stewart, Neubert, & Mount, 1998; Heerkens, 2002; Wysocki, 2003).

* Corresponding author at: ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Campus Universitario, Paraje Arroyo Seco, Tandil 7000, Buenos Aires, Argentina. Tel.: +54 (0249) 4439682x28; fax: +54 (0249) 4439681x52.
E-mail addresses: vyannibe@exa.unicen.edu.ar (V. Yannibelli), amandi@exa.unicen.edu.ar (A. Amandi).

0957-4174/$ - see front matter © 2012 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2012.10.058

Unlike the above-mentioned problems, the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) 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 multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a). This is because, in real multi-objective project scheduling problems, the human resources usually have different effectiveness levels in relation to different work contexts (Barrick et al., 1998; Heerkens, 2002; Wysocki, 2003) and, therefore, the effectiveness of a human resource needs to be considered in relation to its work context. To the best of the authors' knowledge, the influence of the work context on the effectiveness of the human resources has not been considered in other multi-objective project scheduling problems. Because of this, it is considered that the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) is really valuable in comparison with other multi-objective project scheduling problems.

In this paper, the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) is addressed. This problem considers two conflicting, priority optimization objectives for project managers. One of the objectives entails minimizing the project makespan, whereas the other objective involves assigning the most effective set of human resources to each project activity. As was previously mentioned, the addressed problem considers that the effectiveness of a human resource depends on various factors inherent to its work context.

To solve the addressed problem, a multi-objective hybrid algorithm is proposed. This is a search and optimization algorithm composed by two search and optimization algorithms: a multi-objective simulated annealing algorithm and a multi-objective evolutionary algorithm. The multi-objective simulated annealing algorithm is integrated into the framework of the multi-objective evolutionary algorithm. This is meant to improve the performance of the evolutionary-based search (Coello Coello, Lamont, & Veldhuizen, 2007; Corchado, Abraham, & Carvalho, 2010; Corchado, Graña, & Wozniak, 2012; Deb, 2009; Ishibuchi, Yoshida, & Murata, 2003). Specifically, the multi-objective simulated annealing algorithm serves two purposes. At the early stages of the evolutionary-based search, when this search is diverse, the simulated annealing algorithm behaves like an exploitation process to fine-tune the solutions reached by the evolutionary-based search. At later stages of the evolutionary-based search, when this search starts to converge, the simulated annealing algorithm behaves like an exploration process to diversify the solutions reached by the evolutionary-based search and thus to allow this search progresses. To achieve the two mentioned purposes, the behavior of the multi-objective simulated annealing algorithm is self-adaptive based on observations from the state of the evolutionary-based search.

The multi-objective hybrid algorithm generates an approximation to the true Pareto set as a solution to the problem. Thus, the algorithm can provide a number of solutions (i.e., project schedules) with different trade-offs between the optimization objectives to project managers.

A multi-objective hybrid algorithm is proposed because of the following reasons. The problem addressed here can be seen as a multi-objective case of the RCPSP (Resource Constrained Project Scheduling Problem) (Blazewicz, Lenstra, & Rinnooy Kan, 1983) and, therefore, the problem is an *NP-Hard* problem. In this sense, the hybridization of multi-objective evolutionary algorithms with other search and optimization techniques (e.g., simulated annealing) has been proven to be more effective than the classical multi-objective evolutionary algorithms in the resolution of a wide variety of multi-objective *NP-Hard* problems and, in particular, in the resolution of different kinds of multi-objective scheduling problems (Coello Coello et al., 2007; Corchado et al., 2012; Deb, 2009; Ishibuchi et al., 2003, 2010). Thus, it is considered that a multi-objective hybrid algorithm could outperform the multi-objective evolutionary algorithm presented in Yannibelli and Amandi (2012a) for solving the addressed problem. The multi-objective evolutionary algorithm presented in Yannibelli and Amandi (2012a) is the only multi-objective algorithm previously proposed in the literature for solving the addressed problem.

The remainder of the paper is organized as follows. In Section 2, a brief review of published works that consider the effectiveness of human resources in the context of multi-objective project scheduling problems is given. In Section 3, the addressed multi-objective project scheduling problem is described. In Section 4, the multi-objective hybrid algorithm designed to solve the problem is described. In Section 5, the computational experiments developed to evaluate the performance of the multi-objective hybrid algorithm are presented, and their results are analyzed. Finally, in Section 6, the conclusions of the present work are presented.

2. Related works

In the literature, various multi-objective project scheduling problems have considered the effectiveness of human resources. These multi-objective project scheduling problems state different assumptions about the effectiveness of human resources. In this regard, only few multi-objective project scheduling problems have considered human resources with different levels of effectiveness (Bellenguez & Néron, 2005; Gutjahr et al., 2008; Hanne & Nickel, 2005; Yannibelli & Amandi, 2012a), a central aspect in real multi-objective project scheduling problems. In this section, the attention is focused on analyzing the way in which the effectiveness of human resources is considered in multi-objective project scheduling problems reported in the literature.

Li and Womer (2009), Drezet and Billaut (2008) and Bellenguez (2008) address multi-skill project scheduling problems considering different optimization objectives. 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).

Bellenguez and Néron (2005) consider 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.

Hanne and Nickel (2005) address a multi-skill project scheduling problem considering different optimization objectives. In this problem, most activities require only one employee with a particular skill, and each available employee masters different skills. In addition, 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.

Aickelin, Burke, and Li (2009) and Valls, Pérez, and Quintanilla (2009) address the skilled workforce project scheduling problem considering different optimization objectives. In this problem, each project activity requires only one employee with a particular skill, and each available employee has different skills. In Valls et al. (2009), given a skill, for each employee that masters the skill, an efficiency level is defined (heterogeneous efficiencies in relation to each skill). In Aickelin et al. (2009), employees with homogeneous efficiencies in relation to each skill are considered. The two works consider employees with homogeneous levels of effectiveness in relation to each skill.

Gutjahr et al. (2008) and Heimerl and Kolisch (2010) address the problem of scheduling multiple projects taking into account different optimization objectives. Gutjahr et al. (2008) consider human resources with different levels of effectiveness and heterogeneous efficiencies in relation to each skill. Heimerl and Kolisch (2010) consider human resources with homogeneous levels of effectiveness and heterogeneous efficiencies in relation to each skill.

In contrast with the above-mentioned problems, the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) 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 multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a). This is because, in real multi-objective project scheduling problems, the human resources have different effectiveness levels in relation to different work contexts (Barrick et al., 1998; Heerkens, 2002; Wysocki, 2003). Therefore, the effectiveness of a human resource needs to be considered in relation to its work context. To the best of the authors' knowledge, the influence of the work context on the effectiveness of the human resources has not been considered in other multi-objective project scheduling problems. For this reason, it is considered that the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) is really valuable in comparison with other multi-objective project scheduling problems.

3. Multi-objective project scheduling problem description

In this paper, the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) is addressed. This problem is a multi-objective extension of the single-objective project scheduling problem described in Yannibelli and Amandi (2011), Yannibelli and Amandi (2012b).

A description of the multi-objective project scheduling problem addressed in the current paper is presented below.

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, that is to say, when an activity starts, it must be developed period by period until it is completed (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. A skill is considered to be a specialization in a knowledge area.

It is considered that companies and organizations 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, different work contexts can be defined 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 experiences, the labor relations between r and the other employees of g, his or her knowledge, his or her studies, his or her 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 objectives are reached. In this sense, two priority objectives are considered for project managers at the early stage of the project schedule design. One objective is that the most effective set of employees be assigned each project activity. The other objective is to minimize the project makespan. The first objective is modeled by Eqs. (1) and (2), and the second objective is modeled by Eqs. (3) and (4).

Eq. (1) maximizes the effectiveness of the sets of employees assigned to the N activities of a given project. In this equation, 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 $R(j,s)$.

Eq. (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 $R(j,s)$. The mean effectiveness level is used because of the reasons presented below. It is considered that the sets of employees are assigned to project activities with the following properties. First, in the activities considered here, the effectiveness level of a set of employees depends on the effectiveness level of each employee belonging to the set. Second, the higher the sum of the effectiveness levels of those employees, the higher the effectiveness of the set. In the case of project activities with the properties mentioned, it is considered that the mean effectiveness level of the employees of a set is a good predictor of the effectiveness of the set. This is because the mean effectiveness level of the employees of a set is directly proportional to the sum of the effectiveness levels of those employees. Specifically, the higher the sum of the effectiveness levels of the employees, the higher the mean effectiveness level. Thus, if the mean effectiveness level is used as a predictor, the higher the sum of the effectiveness levels of the employees, the higher the effectiveness of the set.

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

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

Eq. (3) minimizes the makespan of a given project. In this equation, set S contains all the feasible schedules for the project in question. The term $d(s)$ represents the makespan of schedule s.

In Eq. (4), the finish times defined by schedule s for N project activities are considered, and $d(s)$ is determined by the maximal finish time. The term $ft(j,s)$ represents the finish time of activity j in schedule s, and the term $st(j,s)$ represents the start time of j in s. Then, $ft(j,s)$ is calculated as $st(j,s)$ plus the duration of j.

$$\min_{s\in S}(d(s)) \tag{3}$$

$$d(s)=\max_{j=1\text{ to }N}(ft(j,s)=st(j,s)+d_j) \tag{4}$$

The two above-described optimization objectives are considered to be possibly conflicting. When the objective of minimizing the project makespan is taken into account, the effectiveness levels of the sets of employees are no considered. Therefore, an optimal schedule in relation to the project makespan could define sets of employees having effectiveness levels lower than the optimal levels. On the other hand, when the objective of maximizing the effectiveness of the sets of employees is taken into account, the project makespan is not considered. Thus, an optimal schedule in relation to the effectiveness levels of the sets of employees could define a project makespan longer than the optimal makespan. Based on the mentioned, there may not be a solution which optimizes all objectives at the same time. In this respect, the Pareto set, or an approximation to the Pareto set, is considered as a solution to the multi-objective problem. The Pareto set is the set of all non-dominated solutions. The concept of dominance is considered as follows. Given two solutions, both of which have scores according to some set of objective values, one solution is considered to dominate the other if its score is better or equal for all objectives, and is strictly better for at least one. The scores that a solution x gets for n objectives are represented as a n-dimensional vector $\bar{x}$. Using the $\succ$ symbol to indicate domination, the relation $x \succ y$ is formally presented in Eq. (5).

$$x \succ y \iff \forall i \in \{1,\ldots,n\}\ x_i \text{ is better than or equal to } y_i, \text{ and } \exists i \in \{1,\ldots,n\}\ x_i \text{ is better than } y_i. \tag{5}$$

For conflicting objectives, there is no single solution dominating all others, and a solution is considered non-dominated if it is not dominated by any other. The set of all non-dominated solutions is considered the Pareto Set.

The Pareto set is formally presented in Eq. (6). In this equation, the term S represents the set of all feasible solutions.

$$\text{Pareto Set} = \{x \in S : \exists y \in S : y \succ x\} \tag{6}$$

In the problem addressed here, it is considered that once the Pareto set is determined, or an approximation to the Pareto set (i.e., a number of near non-dominated solutions) is calculated, the project manager can select the solution that he or she prefers out of the set of solutions obtained. To make this selection, it is considered that the project manager should define his or her preferences about the trade-off between the optimization objectives, and then he or she should choose the solution(s) that closely match the desired trade-off.

For a more detailed discussion of the problem addressed here, readers are referred to the work (Yannibelli & Amandi, 2012a) that has introduced this problem. In the mentioned work, the problem has been introduced and described in detail. Moreover, the mentioned work contains a detailed example of the problem and a feasible solution for this example.

4. A multi-objective hybrid algorithm: integrating a multi-objective simulated annealing algorithm and a multi-objective evolutionary algorithm

To solve the addressed problem, a multi-objective hybrid algorithm is proposed. This is a search and optimization algorithm composed by two search and optimization algorithms: a multi-objective simulated annealing algorithm and a multi-objective evolutionary algorithm. The multi-objective simulated annealing algorithm is integrated into the multi-objective evolutionary algorithm. This is meant to improve the performance of the evolutionary-based search (Coello Coello et al., 2007; Corchado et al., 2010, 2012; Deb, 2009; Ishibuchi et al., 2003). Specifically, the multi-objective simulated annealing algorithm pursues two aims. At the early stages of the evolutionary-based search, when the population of the evolutionary algorithm is diverse, the simulated annealing algorithm behaves like an exploitation process to fine-tune the solutions in the population. At later stages of the evolutionary-based search, when the population of the evolutionary algorithm starts to converge or when the evolutionary-based search is stagnated, the simulated annealing algorithm behaves like an exploration process to introduce diversity in the population of solutions and thus to prevent the premature convergence of the evolutionary-based search. To achieve the two mentioned aims, the behavior of the multi-objective simulated annealing algorithm is self-adaptive based on the diversity level of the population of the underlying multi-objective evolutionary algorithm.

4.1. General behavior of the multi-objective hybrid algorithm

Fig. 1 describes the general behavior of the multi-objective hybrid algorithm.

As seen in Fig. 1, the multi-objective hybrid algorithm is an iterative process. This process starts from an initial population of solutions. Each solution of this initial population encodes a feasible

Multi-objective hybrid algorithm

inputs: population_size, number_generations, P_c, P_m, number_iterations, α (cooling factor)
outputs: non_dominated_solution_set

procedure:
1: population = generate_initial_population(population_size);
2: generation = 1;
3: **while** (generation ≤ number_generations) **do**
4: multi_objective_simulated_annealing_stage(population, number_iterations, α);
5: mating_pool = parent_selection_process(population);
6: offprings = crossover_process(mating_pool, P_c);
7: mutation_process(offsprings, P_m);
8: population = survival_selection_process(population,offprings);
9: generation = generation + 1;
10: **end while**
11: non_dominated_solution_set = get_non_dominated_solutions_from(population);
12: **return** non_dominated_solution_set;

Fig. 1. Description of the multi-objective hybrid algorithm.

schedule for the project to be scheduled. Moreover, each solution has a fitness value that represents the quality of the related schedule with respect of the two optimization objectives considered as part of the multi-objective project scheduling problem. The iterative process ends when a number of generations is reached. After this happens, the iterative process provides the non-dominated solution set of the last population or generation as a solution to the multi-objective project scheduling problem.

In each iteration, the multi-objective hybrid algorithm develops the following steps. First, a multi-objective simulated annealing algorithm is applied to each solution of the current population. This algorithm behaves like either an exploitation process or an exploration process depending on the diversity level of the current population. Thus, the algorithm modifies the solutions of the current population.

Then, 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. The behavior of the mutation process depends on a probability P_m. The mutation process is applied with the aim of introducing diversity in the new solutions generated by the crossover process.

Finally, a survival selection process 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.

4.2. Components of the multi-objective hybrid algorithm

In the next sections, the main components of the multi-objective hybrid algorithm are described. These components are the encoding and decoding of solutions, the multi-objective fitness function, the multi-objective simulated annealing algorithm, and the parent selection, crossover, mutation and survival selection processes.

4.2.1. Encoding and decoding of solutions

To encode or represent the solutions of the population, the encoding described in Yannibelli and Amandi (2012a) for project schedules was used. Using this encoding, each solution is encoded by two lists having as many positions as activities in the project to be scheduled.

The first list is a classical activity list. Each position i on this list contains a different activity j of the project. Each activity j can appear on the list in any position higher than the positions of all its predecessors. Thus, the activity list is a feasible precedence list of the activities involved in the project.

The second list is an assigned resources list. This list details 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.

To build the schedule related to the above-described encoding, the serial schedule generation method (Kolisch & Hartmann, 1999) was used. This method incorporates the project activities in the schedule according to the order given by the activity list. When an activity must be incorporated in the schedule, the method defines the earliest feasible starting time for this activity. In this respect, the method considers that an activity can start after all the predecessors of the activity have been completed and when all the employees assigned to the activity are available. Thus, the schedule built by the method is always a feasible one.

In relation to the behavior of the above-described serial schedule generation method, note that when this method is applied, only one schedule can be built from a given encoded solution, but different encoded solutions could be transformed in the same schedule (Kolisch & Hartmann, 1999).

In relation to the generation of the encoded solutions of the initial population, the random-based generation process described in Yannibelli and Amandi (2012a) was used. By this process, a diverse initial population is obtained. This is meant to avoid the early stagnation of the search developed by the multi-objective hybrid algorithm.

4.2.2. Multi-objective fitness function

This function is used by different components of the multi-objective hybrid algorithm (i.e., multi-objective simulated annealing algorithm, parent selection, survival selection) to determine the fitness values of the encoded solutions. The fitness value of an encoded solution represents the quality of the related schedule with respect of the two optimization objectives considered as part of the addressed problem. One objective is to minimize the project makespan, while the other is to maximize the effectiveness level of the sets of employees assigned to the project activities. Therefore, the multi-objective fitness function evaluates a given encoded solution in relation to each one of the two mentioned optimization objectives and then defines a scalar fitness value for the solution based on the results obtained by the evaluations.

The detailed behavior of the multi-objective fitness function is described as follows. Considering a given encoded solution, the

function decodes the schedule s related to the solution by using the serial method described in Section 4.2.1. Then, the function calculates the value of the term $e(s)$ corresponding to s (Eqs. (1) and (2)), and calculates the value of the term $d(s)$ corresponding to s (Eqs. (3) and (4)).

In order to calculate the value of the term $e(s)$, the function utilizes the values of the terms $e_{rCr,j,k,g}$ inherent to s (Eq. (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. Note that the term $e(s)$ takes a real value over $[0,\ldots,N]$.

In order to calculate the value of the term $d(s)$, the fitness function considers the values of the terms $ft(j,s)$ inherent to s (Eq. (4)).

Once the terms $e(s)$ and $d(s)$ corresponding to schedule s have been calculated, the fitness function defines a scalar fitness value for s based on the mentioned terms. To define a scalar fitness value for s, the fitness function uses the dominance grade approach (Coello Coello et al., 2007; Deb, 2009).

The dominance grade approach is one of the most used approaches to define a scalar fitness value on the basis of a set of values corresponding to different optimization objectives (Coello Coello et al., 2007; Deb, 2009; Eiben & Smith, 2007). This approach has been effectively used in previous works that propose multi-objective evolutionary algorithms to solve multi-objective project scheduling problems (Hanne & Nickel, 2005; Yannibelli & Amandi, 2012a). It also has been effectively used in previous works that propose multi-objective evolutionary algorithms to solve other kinds of multi-objective scheduling problems (Lau et al., 2009).

The dominance grade approach defines a scalar fitness value on the basis of information about the dominance of the given solution. This information is used with the aim of discovering the non-dominated solutions. Specifically, the dominance grade approach assigns to each solution a fitness value equal to the solution's dominance grade (Coello Coello et al., 2007; Deb, 2009; Konak, Coit, & Smith, 2006).

The dominance grade of a solution refers to the number of solutions that it dominates. Formally, the dominance grade of a solution x is given by Eq. (7). In this equation, the term P represents the population on which the dominance grade is calculated. The value of the term P depends on the process that calls the fitness function. In this respect, the multi-objective hybrid algorithm has three processes that call the fitness function and use the values returned by this function.

The first of the three above-mentioned processes is the multi-objective simulated annealing algorithm. The second process is the parent selection. The third process is the survival selection. The multi-objective simulated annealing algorithm is applied to the solutions of the current population. Thus, in this case, the term P is equal to the current population. The parent selection is developed on the population obtained by the multi-objective simulated annealing stage. Therefore, in this case, the term P is equal to the population obtained by the multi-objective simulated annealing stage. The survival selection is developed on the population obtained by the multi-objective simulated annealing stage and on the solution set generated from the mating pool. Thus, in this case, the term P is equal to the set composed of the solutions obtained by the multi-objective simulated annealing stage and the solutions generated from the mating pool.

$$domgrade(x) = |\{y \in P : x \succ y\}| \tag{7}$$

4.2.3. Multi-objective simulated annealing algorithm

In each iteration of the multi-objective hybrid algorithm, a multi-objective simulated annealing stage is applied to the current population. Specifically, a multi-objective simulated annealing algorithm is applied to each solution of the current population. This algorithm behaves like either an exploitation process or an exploration process depending on the diversity level of the current population. Thus, the multi-objective simulated annealing algorithm modifies the solutions of the current population according the diversity of this population.

Fig. 2 describes the multi-objective simulated annealing stage of the multi-objective hybrid algorithm. Fig. 3 describes the general behavior of the multi-objective simulated annealing algorithm.

As seen in Fig. 3, the behavior of the multi-objective simulated annealing algorithm is adapted to either an exploitation or exploration behavior by the temperature parameter. The temperature determines the probability of accepting new solutions that are worse than the current solution. If the temperature is high, the acceptance probability is also high, and vice versa. In the algorithm, the temperature is inversely proportional to the diversity of the current population. Such diversity is represented by the spread of fitnesses within the current population. Based on the mentioned, when the diversity of the current population

Multi-objective simulated annealing stage

inputs: population, number_iterations, α (cooling factor)
outputs: /* population is a set of solutions to which the multi-objective simulated annealing algorithm will be applied */

```
procedure:
1:  max_fitness = maximal_fitness(population);
2:  min_fitness = minimal_fitness(population);
3:  diversity = |max_fitness − min_fitness|;
4:  temperature = 1 / diversity;
5:  new_population = create_empty_population();
6:  i = 1;
7:  while (i ≤ size(population)) do
8:      solution = get_solution(population, i);
9:      if ( fitness(solution, population) ≠ max_fitness ) then
10:         new_solution = multi_objective_simulated_annealing_algorithm(solution, temperature,
                                                                          number_iterations, α,
                                                                          population);
11:     else
12:         new_solution = solution;
13:     end if
14:     add_solution(new_population, new_solution);
15:     i = i + 1;
16: end while
17: population = new_population;
```

Fig. 2. Description of the multi-objective simulated annealing stage of the multi-objective hybrid algorithm.

Multi-objective simulated annealing algorithm

inputs: solution, temperature, number_iterations, α (cooling factor), population
outputs: new_solution

```
procedure:
 1: s = solution;
 2: t = temperature;
 3: i = 1;
 4: while ( (t > 0) and (i ≤ number_iterations) ) do
 5:     new_s = generate_new_solution_from(s);
 6:     if ( fitness(s, population) < fitness(new_s, population)) then
 7:         s = new_s;
 8:     else
 9:         delta = fitness(s, population) – fitness(new_s, population);
10:         acceptance_probability = exp(-delta / t);
11:         x = random(0, 1)
12:         if (x < acceptance_probability) then
13:             s = new_s;
14:         end if
15:     end if
16:     t = t ×α;
17:     i = i + 1;
18: end while
19: new_solution = s;
20: return new_solution;
```

Fig. 3. Description of the multi-objective simulated annealing algorithm.

converges, the temperature rises and then the probability of accepting adverse moves also rises. A consequence of this is that the algorithm will be able to move away from the solution to which it is applied, exploring different basins of attraction of the search space. Eventually, the diversity of the current population will increase lowering the temperature of this population.

Note that a new solution generated from the current solution is worse than the current solution when the fitness of the new solution is lower than the fitness of the current solution. To determine the fitness of a given solution, the algorithm uses the multi-objective fitness function described in Section 4.2.2. Thus, the fitness of a given solution is determined in respect of the two optimization objectives considered as part of the addressed problem.

In relation to the generation of a new solution from the current solution, the algorithm uses a move operator specially designed. This move operator works as follows. Considering a given solution, a move operation for activity lists is applied to the activity list of the given solution. The result of this operation consists of a new activity list. Then, a move operation for assigned resources lists is applied to the assigned resources list of the given solution. The result of this operation consists of a new assigned resources list. Thus, the move operator generates the components of a new solution from the given solution.

In relation to the move operation for activity lists, the simple shift operation for activity lists described in Kolisch and Hartmann (1999) is used. Given an activity list, this operation randomly selects only one activity in the list and moves it from its position to a new feasible position selected randomly.

In relation to the move operation for assigned resources lists, a move operation that is a special case of the classical random resetting (Eiben & Smith, 2007) is used. Given an assigned resources list, this operation randomly selects only one position of the list and replaces the resource assignment on this position by a new feasible resource assignment selected randomly.

4.2.4. Parent selection process

The parent selection process is used to determine which solutions of the population will compose the mating pool. This process is really relevant because the solutions in the mating pool, called parent solutions, will be used by the crossover process in order to generate new solutions, called offspring solutions.

In this work, the parent selection process called tournament selection (Eiben & Smith, 2007) with a tournament size equal to two was applied. By using this process, the solutions with the greatest fitness values within the population will have more chances of being incorporated in the mating pool.

The tournament selection process, with a tournament size equal to two, works as follows. Two solutions are randomly selected from the population and compete for being incorporated in the mating pool. The better one (i.e., the solution with the highest fitness value) is incorporated into the mating pool. Then, both solutions are returned to the population. This operation is repeated until a number M of solutions is incorporated in the mating pool, where M is the population size.

4.2.5. Crossover process

The solutions in the mating pool are paired considering the order in which they were incorporated in the mating pool. Then, a crossover operation is applied to each of these pairs of solutions with a predefined probability P_c to generate new solutions. Specifically, the crossover operation applied to a pair of solutions, called parent solutions, combines the characteristics of these solutions and generates two new solutions, called offspring solutions. Thus, the crossover operation has the possibility of combining the best characteristics of the parent solutions so that new, better solutions can be defined (Eiben & Smith, 2007; Goldberg, 2007). The behavior of the crossover operation used in this work is described below.

Given two parent solutions (parent 1 and parent 2) that must be recombined, the crossover operation works as follows. First, a crossover operation for activity lists is applied to the activity lists of the parent solutions. The result of this operation consists of two new activity lists. The first one is assigned to the first offspring (offspring 1) and the second one is assigned to the second offspring (offspring 2). Then, a crossover operation for assigned resources lists is applied to the assigned resources lists of the parent solutions. The result of this operation consists of two new assigned resources lists. The first one is assigned to the first offspring while the second one is assigned to the second offspring. Thus, two new solutions (offspring 1 and offspring 2) are generated from the two parent solutions (parent 1 and parent 2).

In relation to the crossover operation for activity lists, this operation works as follows. Given the activity lists of parent 1 and parent 2, the operation defines two random crossover points $c1$ and $c2$, considering $1 \leqslant c1 < c2 < N$. Then, the first $c1$ activities on the list of parent 1 are positioned in the first $c1$ positions on the list of offspring 1, in the same order. Subsequently, the operation goes through the activity list of parent 2 and selects the first $(c2 - c1)$ activities not included in the list of offspring 1. Then, the operation copies these activities in the positions $[c1 + 1, c2]$ of the list of offspring 1. The activities are copied considering the order in which they appear in the list of parent 2. Finally, the activities not yet included in the list of offspring 1 are positioned in the empty positions of this list. These activities are positioned taking into account the order in which they appear in the list of parent 1. The activity list generated for offspring 1 is a precedence feasible list.

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

The above-described crossover operation for activity lists corresponds to the two-point crossover developed by Hartmann (1998) for activity lists.

In relation to the crossover operation for assigned resources lists, this operation works as follows. Given the assigned resources lists of parent 1 and parent 2, the operation generates a string V of N random values from a uniform distribution over $[0,1]$, $V = \langle v_1, \ldots, v_p, \ldots, v_N \rangle$. Then, for all positions $p = 1, \ldots, N$, if $v_p \leqslant 0.5$, the

resource assignment for position p in the list of offspring 1 (in the list of offspring 2) is inherited from parent 1 (from parent 2). On the other hand, if $v_p > 0.5$, the resource assignment for position p in the list of offspring 1 (in the list of offspring 2) is inherited from parent 2 (from parent 1). This operation always leads to new feasible assigned resources lists.

The above-described crossover operation for assigned resources lists corresponds to the standard uniform crossover (Eiben & Smith, 2007).

4.2.6. Mutation process

The previously described crossover process generates a new set of solutions from the mating pool. Subsequently, a mutation operation is applied to each solution of this new set in order to randomly modify one or more characteristics of some of these solutions and thus to introduce genetic diversity in the set (Eiben & Smith, 2007; Goldberg, 2007). The behavior of the mutation operation used in this work is described below.

Given a solution that must be mutated, the mutation operation works as follows. First, a mutation operation for activity lists is applied to the activity list of the given solution. The result of this operation consists of a new activity list. Then, a mutation operation for assigned resources lists is applied to the assigned resources list of the given solution. The result of this operation consists of a new assigned resources list. Thus, a mutated solution is generated from the given solution.

In relation to the mutation operation for activity lists, the mutation operation proposed by Hartmann (1998) for activity lists was used. Given an activity list, this operation works as follows. For all positions $p = 1,\ldots,N-1$, the activities on positions p and $p+1$ are exchanged with probability P_m if both activities are not precedence related. By this operation, only precedence feasible lists are generated.

In relation to the mutation operation for assigned resources lists, the mutation operation called random resetting (Eiben & Smith, 2007) was used. Given an assigned resources list, this operation works as follows. For each position of the assigned resources list, a new feasible resource assignment is randomly defined with a probability of P_m. The described mutation operation always leads to new feasible assigned resources lists.

4.2.7. Survival selection process

The survival selection process is applied in order to determine which solutions from the solutions in the population (i.e., population obtained by the multi-objective simulated annealing stage) and the solutions generated from the mating pool will compose the new population.

In this work, a survival selection process called steady state (Eiben & Smith, 2007) was used. By this process, the best solutions obtained by the multi-objective hybrid algorithm are preserved.

In the steady state process, the best $(M - \lambda)$ solutions of the population and the best λ solutions generated from the mating pool are selected to compose the new population, where M is the population size. To develop this process, the parameter λ was set to a value of $M/2$. This value is one of the most used for the parameter λ (Deb, 2009; Eiben & Smith, 2007).

5. Computational experiments to evaluate the multi-objective hybrid algorithm

In this section, the computational experiments developed to evaluate the performance of the multi-objective hybrid algorithm are described. After that, the results obtained are presented and analyzed. Finally, the performance of the multi-objective hybrid algorithm is compared with that of the multi-objective evolutionary algorithm presented in Yannibelli and Amandi (2012a) for solving the addressed problem. The algorithm presented in Yannibelli and Amandi (2012a) is the only multi-objective algorithm previously proposed in the literature for solving the addressed problem.

Table 1
Characteristics of instance sets.

Instance set	Number of activities to be planned in each instance	Number of possible sets of employees per activity	Number of instances
j30_5	30	1–5	40
j30_10	30	1–10	40
j30_15	30	1–15	40
j60_5	60	1–5	40
j60_10	60	1–10	40
j60_15	60	1–15	40
j120_5	120	1–5	40
j120_10	120	1–10	40
j120_15	120	1–15	40

To develop the computational experiments, the nine instance sets presented in Yannibelli and Amandi (2012a) were used. Table 1 shows the main characteristics of these instance sets. The name of each instance set contains two numbers: the first one indicates the number of activities to be planned in each instance and the second one indicates the maximal number of possible sets of employees per activity. Each set contains 40 instances. Moreover, the sets have no instances in common.

Each of the instances contains information about a number of activities to be planned. For each activity, the instance details the duration, the precedence relations, the required skills, and the number of employees required for each required skill.

Moreover, each instance contains information about the skills available to develop the activities and the employees available for each of these skills. In this respect, each instance considers four available skills and assumes that each available employee masters only one of the four possible skills. Each instance also contains information about the effectiveness level of each available employee r in relation to each of the possible work contexts for r in the instance. Specifically, each instance contains all the terms $e_{rCr,j,k,g}$ inherent to each employee r of the instance and a random value over [0,1] for each of the mentioned terms.

Each instance of the nine instance sets has a known optimal solution in respect to the project makespan and a known optimal solution in respect to the effectiveness level of the sets of employees assigned to the project activities (i.e., the project effectiveness level). These optimal solutions are considered here as references.

The multi-objective hybrid algorithm has been run 20 times on each of the instances of the nine instance sets. As a result of each run, the multi-objective hybrid algorithm provided the non-dominated solution set of the last population or generation.

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

In order to evaluate the performance of the multi-objective hybrid algorithm, some aspects of the quality of the non-dominated solution sets obtained by the experiments were analyzed and the computation times required by the algorithm were also analyzed.

Firstly, the spread and distribution of the solutions in the obtained sets were analyzed. In order to analyze the two mentioned aspects, the spread measure described in Wu and Azarm (2001) and the distribution measure described in Lee, Von Allmen, Fink, Petropoulos and Terrile (2005) were utilized. Based on the developed analysis, the solutions in the obtained sets have a good

Table 2
Parameter setting used for the multi-objective hybrid algorithm.

Parameter	Value
Population Size	50
Number of generations	300
Multi-objective simulated annealing algorithm	
Number of iterations	30
α (cooling factor)	0.9
Crossover process	
Crossover Probability P_c	0.8
Mutation process	
Mutation Probability P_m	0.01

distribution on the trade-off front of the objective space (i.e., the range of values covered by the non-dominated solutions for each optimization objective is varied) and have a good spread on the objective space (i.e., the range of values covered by the non-dominated solutions for each optimization objective is wide). This means that the obtained sets contain solutions with diverse trade-offs between the two optimization objectives.

The accuracy of the solutions in the obtained sets was also analyzed. To analyze this aspect, the accuracy measure utilized in Hanne and Nickel (2005) and Yannibelli and Amandi (2012a) was applied. This measure does not require knowledge of the true non-dominated solution sets of the instances, which are unknown in the current case. The measure only requires a reference value for each optimization objective. In this respect, each instance considered here has a known optimal value for each objective. For each instance, the measure selects the non-dominated solution with the minimal project makespan of each obtained set, and calculates the average percentage deviation of these solutions from the optimal project makespan. Then, the measure selects the non-dominated solution with the maximal project effectiveness level of each obtained set, and calculates the average percentage deviation of these solutions from the optimal project effectiveness level. Finally, for each set of instances, the measure calculates the average value of the average percentage deviations for each optimization objective.

Table 3 presents the average percentage deviation *Av. Dev.* (%) obtained for each instance set in respect of each optimization objective. The second column presents the *Av. Dev.* (%) obtained in respect of the maximization of the project effectiveness level. The third column presents the *Av. Dev.* (%) obtained in respect of the minimization of the project makespan.

In relation to the maximization of the project effectiveness level, the *Av. Dev.* (%) obtained by the multi-objective hybrid algorithm for j30_5, j30_10, j30_15 and j60_5 is 0%. These results indicate that the algorithm has found non-dominated solutions with the optimal project effectiveness level for each instance of each set.

The *Av. Dev.* (%) obtained by the multi-objective hybrid algorithm for j60_10, j60_15, j120_5, j120_10 and j120_15 is greater than 0%. Taking into account that the instances of j60_10 and j60_15, and the instances of j120_5, j120_10 and j120_15 have a known optimal project effectiveness level equal to 60 and 120 respectively, the meaning of the average deviation obtained for each one of these sets was analyzed. In the case of j60_10 and j60_15, average deviations equal to 0.11% and 1.4% indicate that the average level of the non-dominated solutions selected from the sets obtained by the algorithm (i.e., non-dominated solutions with the maximal project effectiveness level) is 59.934 and 59.16 respectively. In the case of j120_5, j120_10 and j120_15, average deviations equal to 0.74%, 0.92% and 2.2% indicate that the average level of the non-dominated solutions selected from the obtained sets is 119.112, 118.896 and 117.36 respectively. Based on the mentioned, it may be stated that the algorithm has obtained non-dominated solutions with very high project effectiveness levels for the instances of j60_10, j60_15, j120_5, j120_10 and j120_15.

Table 3
Av. Dev. (%) obtained by the multi-objective hybrid algorithm for each instance set in respect of each optimization objective.

Instance set	*max* (*e*)	*min* (*d*)
j30_5	0	0.012
j30_10	0	0.008
j30_15	0	0
j60_5	0	0.146
j60_10	0.11	0.07
j60_15	1.4	0.045
j120_5	0.74	0.23
j120_10	0.92	0.17
j120_15	2.2	0.15

In relation to the minimization of the project makespan, the *Av. Dev.* (%) value obtained for j30_15 is equal to 0%. This result indicate that, for each instance of j30_15, the average project makespan of the non-dominated solutions selected from the sets obtained by the algorithm (i.e., non-dominated solutions with the minimal project makespan) is equal to the optimal project makespan. Therefore, it may be stated that the algorithm has obtained non-dominated solutions with optimal makespans for the instances of the set.

The *Av. Dev.* (%) values obtained for j120_5, j120_10, j120_15, j60_5, j60_10, j60_15, j30_5 and j30_10 are lower than 0.24%. These results indicate that, for each instance of the mentioned sets, the average project makespan of the non-dominated solutions selected from the sets obtained by the algorithm is near to the optimal project makespan. Thus, it may be stated that the algorithm has obtained non-dominated solutions with near-optimal makespans for the instances of each mentioned set.

Based on the previous analysis of the average deviations presented in Table 3, the following may be pointed out. For the instances of j30_15, the algorithm has obtained non-dominated solution sets in which: the solution with the maximal project effectiveness level of the set has an optimal project effectiveness level, and the solution with the minimal project makespan of the set has an optimal makespan. For the instances of j30_5, j30_10 and j60_5, the algorithm has obtained non-dominated solution sets in which: the solution with the maximal project effectiveness level of the set has an optimal project effectiveness level, and the solution with the minimal project makespan of the set has a near-optimal makespan. For the instances of j60_10, j60_15, j120_5, j120_10 and j120_15, the algorithm has obtained non-dominated solution sets in which: the solution with the maximal project effectiveness level of the set has a very high project effectiveness level, and the solution with the minimal project makespan of the set has a near-optimal makespan.

The computation times required by the multi-objective hybrid algorithm were also analyzed. Table 4 reports the average computation time in seconds obtained for each instance set. The experiments have been performed on a personal computer Intel Core 2 Duo at 3.00 GHz and 3 GB RAM under Windows XP Professional Version 2002.

In relation to the average computation times presented in Table 4, the following points may be mentioned. For j30_5, j30_10 and j30_15, the average time required by the algorithm was lower than 13 s. For j60_5, j60_10 and j60_15, the average time required by the algorithm was higher than 27 s and lower than 37 s. Then, for j120_5, j120_10 and j120_15, the average time required by the algorithm was higher than 99 s and lower than 156 s. Considering the complexity of the instance sets used, in particular the complexity of the instances of j120_5, j120_10 and j120_15, it may be stated that the average computation times obtained by the multi-objective hybrid algorithm are acceptable.

Table 4
Average computation time (in seconds) obtained by the multi-objective hybrid algorithm for each instance set.

Instance set	Average time (s)
j30_5	8.44
j30_10	10.5
j30_15	12.67
j60_5	27.4
j60_10	31.8
j60_15	36.4
j120_5	99.7
j120_10	121.8
j120_15	155.01

5.1. Comparison with the multi-objective algorithm previously proposed in the literature for the addressed problem

In this section, the performance of the multi-objective hybrid algorithm is compared with that of the multi-objective evolutionary algorithm presented in Yannibelli and Amandi (2012a) for solving the addressed problem. The algorithm presented in Yannibelli and Amandi (2012a) is the only multi-objective algorithm previously proposed in the literature for solving the addressed problem.

For sake of simplicity, the multi-objective evolutionary algorithm presented in Yannibelli and Amandi (2012a) will be referred as algorithm PREA. In contrast with the multi-objective hybrid algorithm, the algorithm PREA is not a hybrid algorithm. In this respect, the framework of the algorithm PREA corresponds to a classical multi-objective evolutionary framework. Thus, the framework of the algorithm PREA only includes the classical evolutionary stages (i.e., parent selection, crossover, mutation, and survival selection).

In Yannibelli and Amandi (2012a), the algorithm PREA was evaluated on the instances of the nine sets previously showed in Table 1 (i.e., sets j30_5, j30_10, j30_15, j60_5, j60_10, j60_15, j120_5, j120_10 and j120_15). Specifically, the algorithm PREA was run 20 times on each of the instances of the nine instance sets. After each run, the algorithm provided the non-dominated solution set of the last population or generation. The 20 non-dominated solution sets obtained by the algorithm PREA for each instance are considered here in order to develop the performance comparison between this algorithm and the multi-objective hybrid algorithm. These sets were provided by the authors of Yannibelli and Amandi (2012a). Moreover, the average percentage deviation *Av. Dev.* (%) obtained by PREA for each instance set in respect of each optimization objective and the average computation time obtained by PREA for each instance set are considered here as detailed in Yannibelli and Amandi (2012a). It is necessary to mention that, as described in Yannibelli and Amandi (2012a), the algorithm PREA was evaluated on a personal computer Intel Core 2 Duo at 3.00 GHz and 3 GB RAM under Windows XP Professional Version 2002. Note that the computer used to evaluate the algorithm PREA has the same characteristics than the computer used here to evaluate the multi-objective hybrid algorithm.

To compare the performance of the multi-objective hybrid algorithm with that of the algorithm PREA, different aspects of the quality of the non-dominated solution sets obtained by the algorithms for each instance set are analyzed and compared in this section. Moreover, the average computation times required by the algorithms for each instance set are analyzed and compared in this section.

In relation to the quality of the non-dominated solution sets obtained by the algorithms for each instance set, the aspects of quality analyzed in this section refer to the size, ratio of non-dominated solutions, accuracy, spread and distribution of the non-dominated solution sets. To analyze these aspects, different performance measures are considered. In this respect, to analyze the accuracy of the non-dominated solution sets, the average percentage deviation *Av. Dev.* (%) for each instance set in respect of each optimization objective is considered. To analyze the other mentioned aspects of the non-dominated solution sets, the measures described below are considered. In the first two of these measures, the term a refers to the multi-objective hybrid algorithm and the term b refers to the algorithm PREA.

To analyze the performance of the algorithms in relation to the size of the obtained non-dominated solution sets (considering that the size of a non-dominated solution set refers to the number of solutions in the set), the following measure is considered. This measure will be referred as $L(a,b)$. For a given instance set, the measure $L(a,b)$ calculates the percentage of instances for which the sizes of the 20 non-dominated solution sets obtained by the algorithm a are larger than the sizes of the 20 sets obtained by the algorithm b. The measure gives a real value over the range $[0,100]$. $L(a,b) = 100$ means that, for all the instances of the given instance set, the sizes of the 20 non-dominated solution sets obtained by the algorithm a are larger than the sizes of the 20 sets obtained by the algorithm b. The opposite $L(a,b) = 0$ means that, for none of the instances of the given instance set, the sizes of the 20 non-dominated solution sets obtained by the algorithm a are larger than the sizes of the 20 sets obtained by the algorithm b. Note that $L(a,b)$ does not have to be equal to $1 - L(b,a)$. Thus, both $L(a,b)$ and $L(b,a)$ are considered here. If $L(a,b) > L(b,a)$, this means that the performance of the algorithm a is better than the performance of the algorithm b in relation to the size of the non-dominated solution sets obtained for the given instance set.

To analyze the ratio of solutions in the sets obtained by the algorithm a that are not dominated by any other solutions in the sets obtained by the algorithm b, the measure described in Ishibuchi et al. (2003) is considered. This measure will be referred as $RNDS(a,b)$. For each instance of a given instance set, the measure $RNDS(a,b)$ considers the 20 non-dominated solution sets obtained by the algorithm a, S_{ta} ($t = 1,\ldots,20$), and considers the 20 non-dominated solution sets obtained by the algorithm b, S_{tb}. Then, for each of the sets S_{ta}, the measure calculates the ratio of solutions in S_{ta} that are not dominated by any other solutions in the set S_{tb} (Eq. (8)). The higher the ratio is, the better the set S_{ta} is. Then, the measure calculates the average value of the ratios obtained for the sets S_{ta}. Thus, the measure obtains an average ratio value for the instance. Finally, for the given instance set, the measure calculates the average value of the ratio values obtained for the instances of the set. The described measure $RNDS(a,b)$ gives a real value over the range $[0,1]$. $RNDS(a,b) = 0$ means that, for all the instances of the given instance set, the solutions in the sets obtained by the algorithm a were dominated by solutions in the sets obtained by the algorithm b. In contrast, $RNDS(a,b) = 1$ means that, for all the instances of the given instance set, the solutions in the sets obtained by the algorithm a were not dominated by any other solutions in the sets obtained by the algorithm b. Note that $RNDS(a,b)$ does not have to be equal to $1 - RNDS(b,a)$. Thus, both $RNDS(a,b)$ and $RNDS(b,a)$ are considered. If $RNDS(a,b) > RNDS(b,a)$, this means that the performance of the algorithm a is better than the performance of the algorithm b in relation to the ratio of non-dominated solutions.

$$RNDS(S_{ta}) = \frac{|S_{ta} - \{x \in S_{ta} : \exists y \in S_{tb} : y \succ x\}|}{|S_{ta}|} \quad (8)$$

To analyze the distribution of the solutions in the sets obtained by each algorithm, the non-uniformity measure described in Lee et al. (2005) is considered. This measure will be referred as *NU*. For each instance of a given instance set, the measure *NU* considers the 20 non-dominated solution sets obtained by the algorithm in

question, S_t ($t = 1,\ldots,20$). For each of the sets S_t, the measure calculates the non-uniformity of the distribution of the solutions on the trade-off front in the objective space. This is calculated by Eq. (9). In this equation, $dist_i$ is the Euclidean distance between consecutive solutions in S_t, $i = 1,\ldots,(|S_t| - 1)$, and $dist_u$ is the average of all distances $dist_i$. Thus, Eq. (9) obtains the standard deviation of the distances $dist_i$ normalized by the average distance $dist_u$. $D(S_t) = 0$ means that the distribution of the solutions on the trade-off front is uniform. The higher the value of $D(S_t)$, the more non-uniform the distribution of the solutions on the trade-off front. Then, the measure *NU* calculates the average value of the non-uniformity values obtained for the sets S_t. Thus, the measure obtains an average non-uniformity value for the instance. Finally, for the given instance set, the measure calculates the average value of the non-uniformity values obtained for the instances of the set. The described measure *NU* gives a real value greater than or equal to 0. $NU = 0$ means that, for the instances of the given instance set, the solutions in the obtained sets have a uniform distribution on the trade-off front. In contrast, $NU > 0$ means that, for at least one of the instances of the given instance set, the solutions in at least one of the obtained sets have a non-uniform distribution on the trade-off front. The higher the value of *NU*, the worst the performance of the algorithm in relation to the distribution of the solutions in the obtained sets. Therefore, a lower value of *NU* is desired.

$$D(S_t) = \sqrt{\frac{\sum_{i=1}^{|S_t|-1} (dist_i/dist_u - 1)^2}{|S_t| - 1}} \tag{9}$$

To analyze the spread of the solutions in the sets obtained by each algorithm, the measure *Overall Pareto Spread* (*OS*) described in Wu and Azarm (2001) is considered. For each instance of a given instance set, the measure *OS* considers the 20 non-dominated solution sets obtained by the algorithm in question, S_t ($t = 1,\ldots,20$). For each of the sets S_t, the measure calculates the ratio between the area of coverage of S_t and the objective space (note that the objective space has two dimensions: the project effectiveness level and the project makespan). This ratio is calculated by Eq. (10). In this equation, the reference solutions P_G and P_B are used to define the size of the objective space. P_G is a good (utopian) solution (i.e., a solution with an optimal project effectiveness level and an optimal project makespan) and P_B is a bad solution (i.e., a solution with a project effectiveness level equal to 0 and a project makespan equal to the sum of the makespans of the project activities). The term M is the number of dimensions of the objective space (i.e., number of objective axes). In the present case, $M = 2$. Eq. (11) calculates, for each objective axis, the ratio between the width of the range of values covered by S_t and the width of the range of values of the objective space. In this equation, $f_m(s)$ refers to the value of solution s in relation to the objective axis m. Note that the higher the value given by Eq. (10), the more wide the spread of S_t. Then, the measure *OS* calculates the average value of the ratios obtained for the sets S_t. Thus, the measure obtains an average ratio for the instance. Finally, for the given instance set, the measure calculates the average value of the ratios obtained for the instances of the set. The described measure *OS* gives a real value over [0,1]. In this respect, the higher the value of the measure *OS*, the more wide the spread of the non-dominated solution sets obtained by the algorithm for the given instance set and the better the performance of the algorithm in relation to the spread of the obtained sets.

$$OS(S_t, P_G, P_B) = \prod_{m=1}^{M} OS_m(S_t, P_G, P_B) \tag{10}$$

$$OS_m(S_t, P_G, P_B) = \frac{|\max_{s \in S_t} f_m(s) - \min_{s \in S_t} f_m(s)|}{|f_m(P_B) - f_m(P_G)|} \tag{11}$$

Tables 5–9 report the performance of the multi-objective hybrid algorithm and the performance of the algorithm PREA in relation to the measures $L(\cdot)$, $RNDS(\cdot)$, *NU*, *OS* and *Av. Dev.* (%), respectively.

Table 5 reports the value obtained by each of the two algorithms in relation to the measure $L(\cdot)$ for each instance set. The lowest value obtained by the multi-objective hybrid algorithm is 70% for j120_15, whereas the value obtained by the algorithm PREA for each instance set is 0%. These values indicate that for at least 70% of the instances of each instance set, the sizes of the 20 non-dominated solution sets obtained by the multi-objective hybrid algorithm are larger than the sizes of the 20 sets obtained by the algorithm PREA. In contrast, for none of the used instances, the sizes of the 20 non-dominated solution sets obtained by the algorithm PREA are larger than the sizes of the 20 sets obtained by the multi-objective hybrid algorithm. Therefore, the multi-objective hybrid algorithm outperformed the algorithm PREA in relation to the size of the obtained non-dominated solution sets.

Table 6 reports the value obtained by each of the two algorithms in relation to the measure $RNDS(\cdot)$ for each instance set. The lowest value obtained by the multi-objective hybrid algorithm is 0.61 for j120_15, whereas the highest value obtained by the algorithm PREA is 0.3 for j30_10. These values indicate that, for the instances of each instance set, at least 61% of the solutions in the sets obtained by the multi-objective hybrid algorithm are not dominated by any other solutions in the sets obtained by the algorithm PREA. In contrast, for the instances of each instance set, at most 30% of the solutions in the sets obtained by the algorithm PREA are not dominated by any other solutions in the sets obtained by the multi-objective hybrid algorithm. Thus, the multi-objective hybrid algorithm outperformed the algorithm PREA in relation to the ratio of non-dominated solutions.

Table 7 reports the value obtained by each of the two algorithms in relation to the measure *NU* for each instance set. For all instance sets, the value obtained by the multi-objective hybrid algorithm is much lower than the value obtained by the algorithm PREA. This means that the distribution of the solutions in the sets obtained by the multi-objective hybrid algorithm on the trade-off

Table 5
Performance of the multi-objective hybrid algorithm and performance of the algorithm PREA in relation to the measure $L(\cdot)$.

Instance set	$L(Hybrid, PREA)$	$L(PREA, Hybrid)$
j30_5	97.5	0
j30_10	95	0
j30_15	95	0
j60_5	90	0
j60_10	85	0
j60_15	82.5	0
j120_5	77.5	0
j120_10	72.5	0
j120_15	70	0

Table 6
Performance of the multi-objective hybrid algorithm and performance of the algorithm PREA in relation to the measure $RNDS(\cdot)$.

Instance set	$RNDS(Hybrid, PREA)$	$RNDS(PREA, Hybrid)$
j30_5	0.86	0.25
j30_10	0.87	0.3
j30_15	0.83	0.28
j60_5	0.74	0.14
j60_10	0.76	0.2
j60_15	0.72	0.18
j120_5	0.62	0.11
j120_10	0.64	0.13
j120_15	0.61	0.1

Table 7
Performance of the multi-objective hybrid algorithm and performance of the algorithm PREA in relation to the measure *NU*.

Instance set	Hybrid algorithm	Algorithm PREA
j30_5	0.056	0.279
j30_10	0.062	0.301
j30_15	0.068	0.293
j60_5	0.081	0.32
j60_10	0.078	0.35
j60_15	0.089	0.386
j120_5	0.09	0.401
j120_10	0.101	0.42
j120_15	0.098	0.418

Table 8
Performance of the multi-objective hybrid algorithm and performance of the algorithm PREA in relation to the measure *OS*.

Instance set	Hybrid algorithm	Algorithm PREA
j30_5	0.53	0.39
j30_10	0.56	0.42
j30_15	0.59	0.45
j60_5	0.5	0.37
j60_10	0.47	0.33
j60_15	0.456	0.3
j120_5	0.42	0.28
j120_10	0.39	0.267
j120_15	0.401	0.25

Table 9
Av. Dev (%) obtained by each of the two algorithms (the multi-objective hybrid algorithm and the algorithm PREA) for each instance set in respect of each optimization objective.

Instance set	Hybrid algorithm		Algorithm PREA	
	max (*e*)	*min* (*d*)	*max* (*e*)	*min* (*d*)
j30_5	0	0.012	0	0.02
j30_10	0	0.008	0	0.012
j30_15	0	0	0.2	0.009
j60_5	0	0.146	0	0.17
j60_10	0.11	0.07	0.4	0.09
j60_15	1.4	0.045	1.8	0.07
j120_5	0.74	0.23	1.1	0.25
j120_10	0.92	0.17	1.25	0.19
j120_15	2.2	0.15	2.7	0.18

front in the objective space is more uniform than the distribution of the solutions in the sets obtained by the algorithm PREA. Specifically, for each dimension of the objective space, the ranges of values covered by the solution sets obtained by the multi-objective hybrid algorithm are more varied than those covered by the solution sets obtained by the algorithm PREA. Therefore, the multi-objective hybrid algorithm outperformed the algorithm PREA in relation to the distribution of the solutions in the obtained sets.

Table 8 reports the value obtained by each of the two algorithms in relation to the measure *OS* for each instance set. For all instance sets, the value obtained by the multi-objective hybrid algorithm is higher than the value obtained by the algorithm PREA. This means that the spread of the non-dominated solution sets obtained by the multi-objective hybrid algorithm is wider than the spread of the non-dominated solution sets obtained by the algorithm PREA. Specifically, for each dimension of the objective space, the ranges of values covered by the solution sets obtained by the multi-objective hybrid algorithm are wider than those covered by the solution sets obtained by the algorithm PREA. Therefore, the multi-objective hybrid algorithm outperformed the algorithm PREA in relation to the spread of the obtained non-dominated solution sets.

Table 9 reports the average percentage deviation *Av. Dev.* (%) obtained by each of the algorithms for each instance set in respect of each optimization objective. Columns 2 and 4 present the *Av. Dev.* (%) obtained for each instance set in respect of the maximization of the project effectiveness level, and columns 3 and 5 present the *Av. Dev.* (%) obtained for each instance set in respect of the minimization of the project makespan. For each instance set, the *Av. Dev.* (%) obtained by the multi-objective hybrid algorithm in respect of the maximization of the project effectiveness level (the minimization of the project makespan) is lower than the *Av. Dev.* (%) obtained by the algorithm PREA in respect of the maximization of the project effectiveness level (the minimization of the project makespan). This means that, compared with the algorithm PREA, the multi-objective hybrid algorithm was more effective to reach non-dominated solutions with an optimal or near-optimal project makespan and non-dominated solutions with an optimal or near-optimal project effectiveness level.

From the analysis of the results presented in Tables 5–9, it may be stated that the quality (i.e., size, ratio of non-dominated solutions, spread, distribution and accuracy) of the non-dominated solution sets obtained by the multi-objective hybrid algorithm is better than the quality of the non-dominated solution sets obtained by the algorithm PREA. In addition, as Table 10 reports, the average computation time required by the multi-objective hybrid algorithm for each instance set is lower than that required by the algorithm PREA. Thus, the multi-objective hybrid algorithm not only found better non-dominated solution sets, but also found them faster. This is mainly because, unlike the algorithm PREA, the multi-objective hybrid algorithm integrates a multi-objective simulated annealing algorithm into the evolutionary framework. At the early evolutionary cycles, the multi-objective simulated annealing algorithm fine-tunes the solutions obtained by the evolutionary search. At later evolutionary cycles, the multi-objective simulated annealing algorithm diversifies the solutions obtained by the evolutionary search to allow this search progresses. Thus, the multi-objective hybrid algorithm can reach better non-dominated solution sets in much less generations than the algorithm PREA for the considered instance sets. In this sense, the multi-objective hybrid algorithm required 300 generations from an initial population with 50 solutions to obtain the reported results, as was mentioned in Table 2, whereas the algorithm PREA required 500 generations from an initial population with 50 solutions, as mentioned in Yannibelli and Amandi (2012a). Thus, although the computation time required by one generation of the multi-objective hybrid algorithm is slightly higher than that required by one generation of the algorithm PREA mainly because of the incorporation of the multi-objective simulated annealing algorithm, the total computation time required by the multi-objective hybrid algorithm is lower than that required by the algorithm PREA.

Based on the above-mentioned, the multi-objective hybrid algorithm may be considered to obtain higher-quality non-dominated solution sets than the algorithm PREA.

6. Conclusions and future work

In this paper, the multi-objective project scheduling problem introduced in Yannibelli and Amandi (2012a) was addressed. This multi-objective project scheduling problem considers two conflicting, priority optimization objectives for managers. One of the objectives entails minimizing the project makespan, whereas the other objective involves assigning the most effective set of human resources to each project activity. This multi-objective project scheduling problem considers that the effectiveness of a human resource depends on various factors inherent to its work context. This is a very important aspect because, in real multi-objective pro-

Table 10
Average computation time (in seconds) obtained by each of the two algorithms (the multi-objective hybrid algorithm and the algorithm PREA) for each instance set. Both algorithms were evaluated on a personal computer Intel Core 2 Duo at 3.00 GHz and 3 GB RAM under Windows XP Professional Version 2002.

Instance set	Hybrid algorithm	Algorithm PREA
j30_5	8.44	10.56
j30_10	10.5	13.2
j30_15	12.67	15.84
j60_5	27.4	34.08
j60_10	31.8	39.76
j60_15	36.4	45.44
j120_5	99.7	124.56
j120_10	121.8	152.24
j120_15	155.01	193.76

ject scheduling problems, the human resources usually have different effectiveness levels in relation to different work contexts (Barrick et al., 1998; Heerkens, 2002; Wysocki, 2003). Therefore, the effectiveness of a human resource needs to be considered in relation to its work context. To the best of the authors' knowledge, the influence of the work context on the effectiveness of the human resources has not been considered in other multi-objective project scheduling problems reported in the literature.

To solve the addressed problem, a multi-objective hybrid algorithm was proposed. This is a search and optimization algorithm composed by two search and optimization algorithms: a multi-objective simulated annealing algorithm and a multi-objective evolutionary algorithm. The multi-objective simulated annealing algorithm is integrated into the framework of the multi-objective evolutionary algorithm. This is meant to improve the performance of the evolutionary-based search. Specifically, the multi-objective simulated annealing algorithm serves two purposes. At the early stages of the evolutionary-based search, when this search is diverse, the simulated annealing algorithm behaves like an exploitation process to fine-tune the solutions reached by the evolutionary-based search. At later stages of the evolutionary-based search, when this search starts to converge, the simulated annealing algorithm behaves like an exploration process to diversify the solutions reached by the evolutionary-based search and thus to allow this search progresses. To achieve the two mentioned purposes, the behavior of the multi-objective simulated annealing algorithm is self-adaptive based on observations from the state of the evolutionary-based search.

The multi-objective hybrid algorithm generates an approximation to the true Pareto set as a solution to the problem. Specifically, the algorithm provides a set of near non-dominated solutions (i.e., project schedules) as a solution to the problem.

To evaluate the performance of the multi-objective hybrid algorithm, the nine instance sets described in Yannibelli and Amandi (2012a) were considered, and the computational experiments developed on these nine instance sets were presented. Then, the performance of the multi-objective hybrid algorithm on the mentioned instance sets was compared with that of the multi-objective evolutionary algorithm previously proposed in Yannibelli and Amandi (2012a) for solving the addressed problem.

In the developed experiments, the multi-objective hybrid algorithm was tested many times on each of the instances of the nine instance sets, and after each one of the runs, the algorithm provided the non-dominated solution set of the last population or generation. To evaluate the performance of the multi-objective hybrid algorithm on each instance set, different aspects of the quality (i.e., spread, distribution and accuracy) of the non-dominated solution sets obtained for each instance set were analyzed. Moreover, the average computation time required by the algorithm for each instance set was analyzed. Based on the developed analysis, it may be stated that the multi-objective hybrid algorithm has obtained non-dominated solution sets with a wide spread, a good distribution and a high accuracy in an acceptable computation time for each instance set.

In order to compare the performance of the multi-objective hybrid algorithm on the nine instance sets with that of the multi-objective evolutionary algorithm previously proposed in Yannibelli and Amandi (2012a), different aspects of the quality (i.e., size, ratio of non-dominated solutions, spread, distribution and accuracy) of the non-dominated solution sets obtained by the algorithms for each instance set were analyzed and compared. Moreover, the average computation times required by the algorithms for each instance set were analyzed and compared. As a result of the comparative analysis conducted, it may be stated that the quality of the non-dominated solution sets obtained by the multi-objective hybrid algorithm is better than the quality of the non-dominated solution sets obtained by the algorithm proposed in Yannibelli and Amandi (2012a). Specifically, the multi-objective hybrid algorithm has obtained non-dominated solution sets with a better size, ratio of non-dominated solutions, spread, distribution and accuracy for each instance set. Moreover, the average computation time required by the multi-objective hybrid algorithm for each instance set is lower than that required by the algorithm proposed in Yannibelli and Amandi (2012a). Thus, the multi-objective hybrid algorithm has been shown to be more effective and efficient than the algorithm proposed in Yannibelli and Amandi (2012a) on the nine instance sets considered here.

Based on the above-mentioned, it is considered that the multi-objective hybrid algorithm proposed in this paper may be used to obtain higher-quality non-dominated solution sets than the algorithm proposed in Yannibelli and Amandi (2012a). Specifically, it is considered that a project manager may obtain a higher number of solutions (i.e., project schedules) with more varied, better trade-offs between the optimization objectives by using the multi-objective hybrid algorithm proposed in this paper.

In the multi-objective hybrid algorithm proposed in this paper, a multi-objective simulated annealing algorithm was integrated like a local exploitation/exploration process within a multi-objective evolutionary algorithm. In future works, the integration of other local exploitation/exploration processes (e.g., multi-objective tabu search and multi-objective hill climbing) within the multi-objective evolutionary algorithm will be investigated. Moreover, other crossover and mutation processes will be evaluated for the encoding of solutions used in the multi-objective hybrid algorithm. In addition, other survival selection processes proposed in the literature (e.g., deterministic crowding) will be examined.

References

Aickelin, U., Burke, E., & Li, J. (2009). An evolutionary squeaky wheel optimization approach to personnel scheduling. *IEEE Transactions on Evolutionary Computation, 13*(2), 433–443.

Barrick, M. R., Stewart, G. L., Neubert, M. J., & Mount, M. K. (1998). Relating member ability and personality to work-team processes and team effectiveness. *Journal of Applied Psychology, 83*(3), 377–391.

Bellenguez, O. (2008). A reactive approach for the multi-skill project scheduling problem. In *PATAT 2008 proceedings of the 7th international conference on the practice and theory of automated timetabling* (pp.18–22) August, Montreal.

Bellenguez, O., & Néron, E. (2005). Lower bounds for the multi-skill project scheduling problem with hierarchical levels of skills. In E. Burke & M. Trick (Eds.). *PATAT 2004, lecture notes in computer science* (Vol. 3616, pp. 229–243). Berlin: Springer.

Blazewicz, J., Lenstra, J., & Rinnooy Kan, A. (1983). Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics, 5*(1), 11–24.

Coello Coello, C. A., Lamont, G. B., & Veldhuizen, D. A. (2007). *Evolutionary algorithms for solving multi-objectives problems* (2nd ed.). 978-0-387-33254-3. New York: Springer.

Corchado, E., Abraham, A., & Carvalho, A. (2010). Hybrid intelligent algorithms and applications. *Information Sciences, 180*(14), 2633–2634.

Corchado, E., Graña, M., & Wozniak, M. (2012). New trends and applications on hybrid artificial intelligence systems. *Neurocomputing, 75*(1), 61–63.

Deb, K. (2009). *Multi-objective optimization using evolutionary algorithms*. 978-0-470-74361-4. New York: Wiley.

Drezet, L. E., & Billaut, J. C. (2008). A project scheduling problem with labour constraints and time-dependent activities requirements. *International Journal of Production Economics, 112*(1), 217–225.

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

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

Gutjahr, W. J., Katzensteiner, S., Reiter, P., Stummer, Ch., & Denk, M. (2008). Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research, 16*(3), 281–306.

Hanne, T., & Nickel, S. (2005). A multiobjective evolutionary algorithm for scheduling and inspection planning in software development projects. *European Journal of Operational Research, 167*(3), 663–678.

Hartmann, S. (1998). A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics, 45*(7), 733–750.

Heerkens, G. R. (2002). *Project management*. USA: McGraw-Hill.

Heimerl, C., & Kolisch, R. (2010). Scheduling and staffing multiple projects with a multi-skilled workforce. *OR Spectrum, 32*(4), 343–368.

Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation, 7*(2), 204–223.

Kolisch, R., & Hartmann, S. (1999). Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In J. Weglarz (Ed.), *Project scheduling: Recent models, algorithms and applications* (pp. 147–178). USA: Kluwer Academic.

Konak, A., Coit, D., & Smith, A. (2006). Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering and System Safety, 91*(9), 992–1007.

Lau, H. C. W., Chan, T. M., Tsui, W. T., Chan, F. T. S., Ho, G. T. S., & Choy, K. L. (2009). A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. *Expert Systems with Applications, 36*(4), 8255–8268.

Lee, S., Von Allmen, P., Fink, W., Petropoulos, A. E., & Terrile, R.J. (2005). Comparison of multi-objective genetic algorithms in optimizing Q-law low-thrust orbit transfers. In *GECCO 2005 (genetic and evolutionary computation conference)* 25–29 June, Washington DC.

Li, H., & Womer, K. (2009). Scheduling projects with multi-skilled personnel by a hybrid MILP/CP benders decomposition algorithm. *Journal of Scheduling, 12*(3), 281–298.

Valls, V., Pérez, A., & Quintanilla, S. (2009). Skilled workforce scheduling in service centers. *European Journal of Operational Research, 193*(3), 791–804.

Wu, J., & Azarm, S. (2001). Metrics for quality assessment of a multiobjective design optimization solution set. *Transactions of the ASME, Journal of Mechanical Design, 123*, 18–25.

Wysocki, R. K. (2003). *Effective project management* (3rd ed.). 0-471-43221-0. USA: Wiley Publishing.

Yannibelli, V., & Amandi, A. (2011). A knowledge-based evolutionary assistant to software development project scheduling. *Expert Systems with Applications, 38*(7), 8403–8413.

Yannibelli, V., & Amandi, A. (2012a). Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan. *Engineering Optimization*. http://dx.doi.org/10.1080/0305215X.2012.658782.

Yannibelli, V., & Amandi, A. (2012b). A memetic approach to project scheduling that maximizes the effectiveness of the human resources assigned to project activities. In E. Corchado et al. (Eds.). *HAIS 2012, part I, lecture notes in computer science* (Vol. 7208, pp. 159–173). Heidelberg: Springer.