Engineering Optimization
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/geno20

Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan

Virginia Yannibelli [a] & Analía Amandi [b]

[a] ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Tandil, Buenos Aires, Argentina

[b] CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

Version of record first published: 18 Apr 2012.

To cite this article: Virginia Yannibelli & Analía Amandi (2013): Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan, Engineering Optimization, 45:1, 45-65

To link to this article: http://dx.doi.org/10.1080/0305215X.2012.658782

Engineering Optimization
Vol. 45, No. 1, January 2013, 45–65

Project scheduling: A multi-objective evolutionary algorithm that optimizes the effectiveness of human resources and the project makespan

Virginia Yannibelli* and Analía Amandi

ISISTAN Research Institute, Fac. Cs. Exactas, UNCPBA, Tandil, Buenos Aires, Argentina, and CONICET, Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

(*Received 23 April 2011; final version received 23 November 2011*)

In this article, the project scheduling problem is addressed in order to assist project managers at the early stage of scheduling. Thus, as part of the problem, two priority optimization objectives for managers at that stage are considered. One of these objectives is to assign the most effective set of human resources to each project activity. The effectiveness of a human resource is considered to depend on its work context. The other objective is to minimize the project makespan. To solve the problem, a multi-objective evolutionary algorithm is proposed. This algorithm designs feasible schedules for a given project and evaluates the designed schedules in relation to each objective. The algorithm generates an approximation to the Pareto set as a solution to the problem. The computational experiments carried out on nine different instance sets are reported.

Keywords: project scheduling; human resource assignment; multi-skilled resources; effectiveness of human resources; multi-objective evolutionary algorithms

1. Introduction

In most organizations, the design of an initial schedule for a given project is a central, non-trivial and costly task. This task involves defining feasible start times and feasible human resource assignments for project activities. 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. In fact, the development and the results of an activity depend on the effectiveness of the resources assigned to it (Heerkens 2002).

Usually, project managers must develop the initial scheduling of a given project in such a way that the optimization objectives defined by the organization are reached. This task requires a considerable amount of time, effort, experience in scheduling and knowledge about the available human resources that can be considered for project activities. Besides, this task poses some risks. Managers may make poor decisions on account of human limitations: lack of knowledge about the organization and its resources, and incorrect assumptions, among others. Therefore, assisting

*Corresponding author. Email: vyannibe@exa.unicen.edu.ar

ISSN 0305-215X print/ISSN 1029-0273 online
© 2013 Taylor & Francis
http://dx.doi.org/10.1080/0305215X.2012.658782
http://www.tandfonline.com

managers at the early stage of project schedule design is a valuable and worthwhile initiative. In this respect, knowledge-based automatic design has the aim of providing initial schedules in an efficient way (*i.e.* time requirements are lower in automatic design than in manual design) and in an effective way (*i.e.* erroneous decisions are minimized on account of all the available knowledge arising from information on already executed projects).

Over the past 30 years, many different kinds of algorithm (*i.e.* exact algorithms, heuristics and metaheuristics) have been proposed in the literature to automatically solve project scheduling problems. In this context, different works have considered specificities of human resources (*i.e.* skills, efficiency, workload capacity and cost per time unit). However, to the best of the authors' knowledge, only a few works have considered human resources with different levels of effectiveness (Bellenguez and Néron 2005, Hanne and Nickel 2005, Gutjahr *et al.* 2008), a central aspect in real projects. These works state assumptions about the effectiveness of the human resources, but these assumptions do not include some aspects that are really significant. In this sense, the works 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 works mentioned above. 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 *et al.* 1998, Heerkens 2002). Therefore, the effectiveness of a human resource should be considered in relation to all the factors mentioned above.

In this article, the project scheduling problem is addressed with the aim of assisting project managers at the early stage of scheduling. Thus, two priority optimization objectives for managers at that stage are considered. One of the objectives involves assigning the most effective set of human resources to each project activity, whereas the other objective entails minimizing the project makespan. These objectives have been considered because the attention is focused on optimizing the development and the results of the project activities as well as the time needed to obtain the final results of the project. In relation to the first mentioned objective, the effectiveness of a human resource is considered to depend on various factors inherent to its work context.

To solve the problem in question, a multi-objective evolutionary algorithm is proposed. Considering a given project, the algorithm designs feasible schedules for the project, and evaluates the designed schedules in relation to each optimization objective. Regarding the first objective, the evaluation is developed on the basis of knowledge about the effectiveness of the human resources involved in each schedule. Regarding the second objective, the evaluation is developed in relation to the makespan of each schedule. As a solution to the problem, the algorithm generates an approximation to the Pareto set.

A multi-objective evolutionary algorithm is proposed because the problem addressed here can be seen as a multi-objective case of the resource constrained project scheduling problem (RCPSP) (Blazewicz *et al.* 1983) and, therefore, the problem is an *NP-hard* problem. In this sense, the multi-objective evolutionary algorithms have been proven to be effective in the resolution of a wide variety of multi-objective *NP-hard* problems and, in particular, in the resolution of multi-objective project scheduling problems (Coello Coello *et al.* 2007, Deb 2009).

The remainder of the article is organized as follows. In Section 2, the problem is defined. In Section 3, a brief review of approaches proposed in literature for project scheduling problems is given in which the effectiveness of human resources is considered. In Section 4, the evolutionary approach designed to solve the problem is described. In Section 5, the computational experiments developed are presented, and their results are analysed. Finally, in Section 6, the conclusions of the present work are presented.

2. Problem description

A project contains a set A of N activities, $A = \{1, \dots, N\}$, 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 due to technological requirements. In general, each of the activities consumes products generated by other activities. Thus, 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.

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.

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

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

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

As a result, it is possible to define different work contexts for each available employee. It is considered that the work context of an employee r, denoted as $C_{r,j,k,g}$, is made up of four main components. The first component refers to the activity j which r is assigned (*i.e.* the complexity of j, its domain, etc.). The second component refers to the skill k which r is assigned within activity j (*i.e.* the tasks associated to k within j). The third component is the set of employees g that has been assigned j and that includes r (*i.e.* r must work in collaboration with the other employees assigned to j). The fourth component refers to the attributes of r (*i.e.* his or her experiences, the labour 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 (Barrick *et al.* 1998, Heerkens 2002, Wysocki 2003). For this reason, it is possible to assume 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 $e_{rC r,j,k,g}$. The term $e_{rC r,j,k,g}$ 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 $e_{rC r,j,k,g}$ takes a real value over the range [0, 1]. The values of the terms $e_{rC r,j,k,g}$ 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 modelled by Equations (1) and (2), and the second objective is modelled by Equations (3) and (4).

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

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

Equation (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 $R(j,s)$.

Equation (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, the mean effectiveness level of the employees of a set is a good predictor of the effectiveness of the set (Barrick *et al.* 1998). 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.

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

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

Equation (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 Equation (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.

The previously defined optimization objectives are possibly conflicting. When the objective of minimizing the project makespan is taken into account, the effectiveness levels of the sets of employees are not 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 these factors, there may not be a solution that optimizes all objectives at the same time. In cases like this, the Pareto set, or an approximation to the Pareto set, is generally considered as a solution to the multi-objective problem (Coello Coello *et al.* 2007, Eiben and Smith 2007, Deb 2009). The Pareto set is the set of all non-dominated solutions. The concept of dominance is described as follows.

Given two solutions, both of which have scores according to some set of objective values, one solution is said 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 can be represented as an n-dimensional vector $\bar{x}$. Using the $\succ$ symbol to indicate domination, the relation $x \succ y$ is formally defined in Equation (5).

$$\begin{aligned} x \succ y \Leftrightarrow \forall i \in \{1,\dots,n\}\ x_i \textit{ is better than or equal to } y_i, \\ \textit{and } \exists i \in \{1,\dots,n\}\ x_i \textit{ is better than } y_i. \end{aligned} \tag{5}$$

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

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

$$\text{Pareto set} = \{x \in S : \nexists\, y \in S : y \succ x\} \tag{6}$$

In the context of the problem addressed here, 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, 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 (Deb 2009).

2.1. *Problem example*

In this section, a brief example of the project scheduling problem addressed here is presented. Then, a feasible schedule for the project of this example is shown.

The project contains 11 activities. Figure 1 shows the precedence relations among the activities of the project and details an estimated duration (expressed in weeks) for each activity.

The project activities require employees with different skills to be developed. In this sense, Table 1 details the skills required for each activity and the number of employees required for each skill.

To develop the complete project, five employees with multiple skills are available. Table 2 details the skills mastered by each one of the five available employees.

Considering the skill requirements of each project activity and the skills of the available employees, it is possible to define more than one feasible set of employees for the activities 1, 2, 6, 10 and 8. For example, it is possible to define three feasible sets of employees for the activity 6 (*i.e.* the sets (E2, E4), (E2, E5) and (E4, E5)).

In relation to the effectiveness level of the available employees, Tables 3–5 show the effectiveness levels of each of employees that can be assigned to the activities 2, 6 and 10, respectively. In each of these tables, for each employee that can be assigned to the activity corresponding to the table, all the possible work contexts inherent to the activity are presented, and an effectiveness

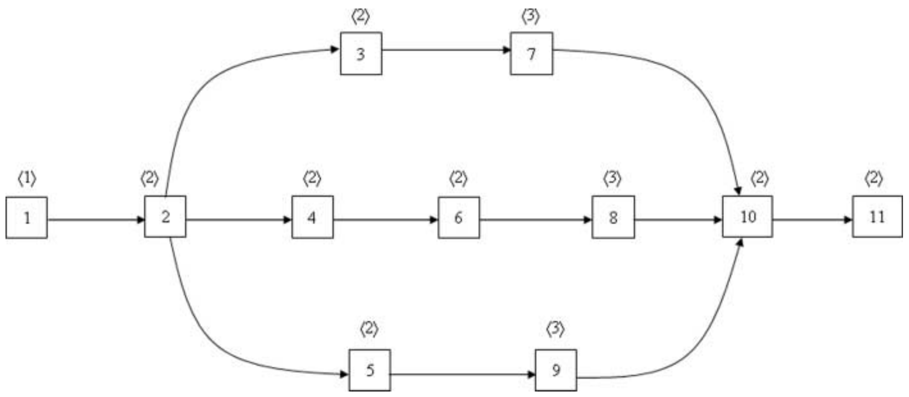


Figure 1. Precedence graph of the project example.

Table 1. Skills required for each activity in the project example.

No. of activity	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5	Skill 6	Skill 7
1	1	1					
2		2					
3				1			
4			1				
5					1		
6						2	
7					1		
8						1	
9					1		
10						2	
11							2

Table 2. Skills mastered by each one of the five available employees in the project example.

Employee	Skill 1	Skill 2	Skill 3	Skill 4	Skill 5	Skill 6	Skill 7
E1	√	√					√
E2		√	√			√	√
E3		√		√	√		
E4						√	
E5						√	

level in relation to each mentioned context is presented. For example, in Table 4, the first column indicates that the employee E2 can be assigned to the activity 6. Then, the two rows corresponding to E2 present two possible work contexts for E2: E2 can be assigned as ‘skill 6’ (*e.g.* designer) to activity 6 working in collaboration with the employee E4 or can be assigned as ‘skill 6’ to activity 6 working in collaboration with the employee E5. Then, the fourth column of the table indicates that, in relation to the first mentioned work context, the effectiveness level of E2 is equal to 0.9 and, in relation to the second mentioned work context, the effectiveness level of E2 is equal to 0.8.

Based on the content of Tables 3–5, and considering the way in which the effectiveness of a set of employees is calculated (Equation 2), it is possible to say that the employees E2 and E4 form the most effective set to develop the activity 6, the employees E2 and E4 also form the most effective set to develop the activity 10, and the employees E2 and E1 form the most effective set to develop the activity 2.

Table 3. Effectiveness levels of each employee that can be assigned to activity 2: for each employee, different effectiveness levels are presented in relation to different work contexts.

Employee	Activity	Skill	Set of employees	Effectiveness level
E1	2	2	(E1, E2)	0.9
	2	2	(E1, E3)	0.7
E2	2	2	(E2, E1)	0.9
	2	2	(E2, E3)	0.8
E3	2	2	(E3, E1)	0.7
	2	2	(E3, E2)	0.8

Table 4. Effectiveness levels of each employee that can be assigned to activity 6: for each employee, different effectiveness levels are presented in relation to different work contexts.

Employee	Activity	Skill	Set of employees	Effectiveness level
E2	6	6	(E2, E4)	0.9
	6	6	(E2, E5)	0.8
E4	6	6	(E4, E2)	0.9
	6	6	(E4, E5)	0.6
E5	6	6	(E5, E2)	0.6
	6	6	(E5, E4)	0.4

Table 5. Effectiveness levels of each employee that can be assigned to activity 10: for each employee, different effectiveness levels are presented in relation to different work contexts.

Employee	Activity	Skill	Set of employees	Effectiveness level
E2	10	6	(E2, E4)	0.8
	10	6	(E2, E5)	0.7
E4	10	6	(E4, E2)	0.8
	10	6	(E4, E5)	0.6
E5	10	6	(E5, E2)	0.5
	10	6	(E5, E4)	0.4

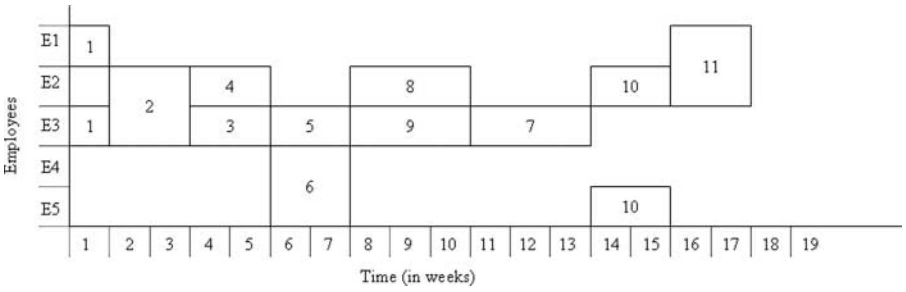


Figure 2. A feasible schedule for the project example.

In addition to the content of Tables 3–5, it is considered that the employee E4 is the most effective to develop the activity 8, and the employees E1 and E2 form the most effective set to develop the activity 1.

Figure 2 shows a feasible schedule for the previously described project. The schedule indicates the specific employees assigned to each activity and the starting time defined for each activity.

3. Related works

Over the past 30 years, a large number of works in the literature have addressed project scheduling problems and other kinds of scheduling problem. Some of these works do not consider specificities of human resources (Kolisch and Hartmann 2006, Chan *et al.* 2006, Cortés Rivera *et al.* 2007, Wong *et al.* 2009, Lau *et al.* 2009), while others do consider different specificities of human resources (Hanne and Nickel 2005, Drezet and Billaut 2008, Li and Womer 2009, Heimerl and Kolisch 2010). In this connection, different works have considered the effectiveness of human resources in the context of project scheduling problems. Nevertheless, a few works have considered human resources with different levels of effectiveness (Bellenguez and Néron 2005, Hanne and Nickel 2005, Gutjahr *et al.* 2008), although with a great number of simplifications. In this section, the attention is focused on analysing the way in which the effectiveness of human resources is considered in related works.

Bellenguez (2008), Bellenguez and Néron (2004, 2007), Néron *et al.* (2006) and Néron (2002) address the multi-skill project scheduling problem. In this problem, 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 the 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.

Li and Womer (2009) address the multi-skill project scheduling problem with the aim of minimizing the total staffing cost, and consider a specific workload capacity (*i.e.* weeks in a project) and a specific salary for each employee. In Drezet and Billaut (2008) the objective of the multi-skill project scheduling problem is to minimize the maximal lateness of the project, and consider a specific workload capacity (*i.e.* duration of work per day, etc.) for each employee. Both works consider homogeneous levels of effectiveness in relation to each skill.

The works mentioned in the preceding two paragraphs consider 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 the multi-skill project scheduling problem with three different optimization objectives (*i.e.* time, quality and cost). In this work, 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.

Valls *et al.* (2007, 2009), Aickelin *et al.* (2009) and Focacci *et al.* (2000) address the skilled workforce project scheduling problem considering different optimization objectives. In this problem, each project activity requires only one worker with a particular skill, and each available worker has different skills. In Valls *et al.* (2007, 2009), given a skill, for each worker that masters the skill, an efficiency level is defined (heterogeneous efficiencies in relation to each skill). In Aickelin *et al.* (2009) and Focacci *et al.* (2000), workers with homogeneous efficiencies in relation to each skill are considered. The four works consider workers with homogeneous levels of effectiveness in relation to each skill.

Heimerl and Kolisch (2010) and Gutjahr *et al.* (2008) address the problem of scheduling multiple projects taking into account different optimization objectives. Heimerl and Kolisch

(2010) consider human resources with homogeneous levels of effectiveness and heterogeneous efficiencies in relation to each skill. They also consider a specific workload capacity and cost per time unit for each resource. Gutjahr *et al.* (2008) consider human resources with different levels of effectiveness and heterogeneous efficiencies in relation to each skill.

In summary, most of the analysed works consider human resources with homogeneous levels of effectiveness in relation to each skill. Some of the analysed works consider human resources with different levels of effectiveness 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. Nevertheless, other contextual factors that also determine the effectiveness of a human resource in a given activity are not considered in the analysed works. Such factors involve the attributes of the activity to which the resource is assigned, the attributes of the skill to which the resource is assigned, the other resources with whom the resource in question must work, as well as the attributes of the resource. In this article, the influence of all the above-mentioned contextual factors on effectiveness of the human resources is considered. Specifically, the effectiveness of a human resource is considered to depend on all the factors mentioned above. Then, for each human resource, it is possible to consider different effectiveness levels in relation to different work contexts. The influence of the work context on the effectiveness of the human resources has not been considered in previous related works.

4. A multi-objective evolutionary algorithm

To solve the problem addressed in this article, a multi-objective evolutionary algorithm is proposed. Multi-objective evolutionary algorithms are adaptive heuristic methods of search and optimization inspired by Darwin's theory of evolution (Eiben and Smith 2007, Goldberg 2007, Coello Coello *et al.* 2007, Deb 2009). According to these algorithms, an initial population of candidate solutions to a problem evolves towards the optimal solutions based on the principles of natural selection, crossover and mutation.

The general behaviour of the algorithm is as follows. Considering a given project, the algorithm starts the evolution from an initial population of solutions in which each solution codifies a feasible project schedule. Then, each solution of the population is decoded (*i.e.* the related schedule is built), and evaluated according to each optimization objective of the problem by a fitness function. As explained earlier, one objective is to maximize the effectiveness of the sets of employees assigned to project activities. In relation to this objective, the fitness function evaluates the assignments of each solution based on available knowledge about the effectiveness of the employees involved in the solution. The other objective is to minimize the project makespan. With regard to this objective, the fitness function evaluates the makespan of each solution. Once the evaluations inherent in the different objectives have been developed, the fitness function defines a scalar fitness value for each solution.

Then, a selection process is used to select a number of solutions from the current population according to a selection strategy. In general, the solutions with the greatest fitness values have more chances of being selected. The selected solutions are paired, and a crossover process is applied to each pair of solutions to generate new feasible ones. Then, a mutation process is applied to modify the components of the solutions generated by the crossover. The purpose of using the mutation process is to promote diversity in the current population of solutions. Finally, the replacement strategy called deterministic crowding (Goldberg 2007) is used to create a new population from the solutions in the current population and the new generated solutions.

This process is repeated until the stopping criterion is reached (*i.e.* a predefined number of iterations or generations). After this happens, the algorithm provides an approximation to the Pareto set as a solution to the problem.

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

4.1. *Representation or encoding of solutions*

In the algorithm, each solution in a population represents a particular project schedule. The solutions must be represented or encoded in such a way that the application of different crossover and mutation operators generates new feasible solutions. Therefore, it is necessary to define an appropriate encoding to project schedules.

A representation based on the standard activity list representation is proposed (Hartmann 1998, Kolisch and Hartmann 1999, 2006). Details about the proposed representation are presented below.

Each solution is represented by two lists having as many positions as activities in the project. Figure 3 shows the proposed representation for a project with N activities.

The first list is a standard activity list. This list is a feasible precedence list of the activities involved in the project (*i.e.* each activity j can appear on the list in any position higher than the positions of all its predecessors). Moreover, each activity j in the list contains information about its development (*i.e.* activity duration and requirement of employees of each skill k).

The second list is an assigned resources list. This list contains information about the employees of every skill k assigned to each activity of the project. Specifically, position j on this list details the employees of every skill k assigned to activity j. The detailed information about the employees assigned to each activity is not considered in the standard activity list representation.

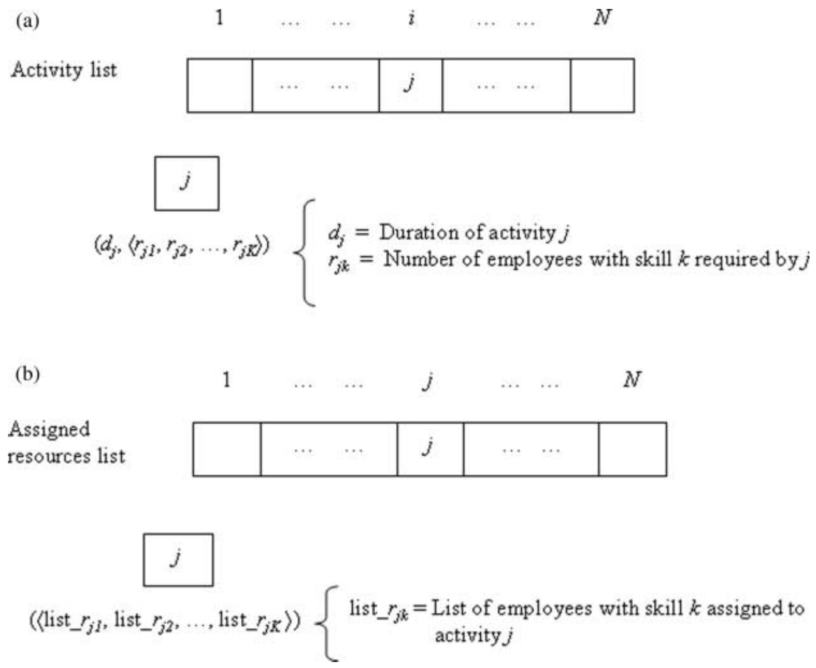
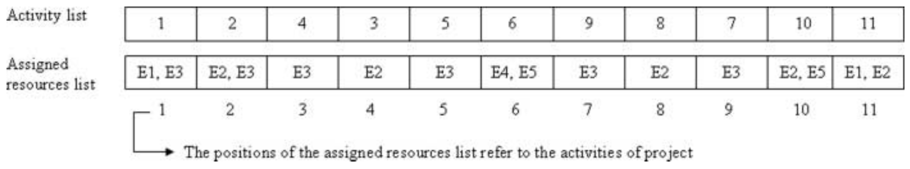

(a)
1 … … i … … N
Activity list
… … j … …
j
$(d_j, \langle r_{j1}, r_{j2}, \ldots, r_{jK}\rangle)$
d_j = Duration of activity j
r_{jk} = Number of employees with skill k required by j
(b)
1 … … j … … N
Assigned resources list
… … j … …
j
$(\langle \text{list\_}r_{j1}, \text{list\_}r_{j2}, \ldots, \text{list\_}r_{jK}\rangle)$
$\text{list\_}r_{jk}$ = List of employees with skill k assigned to activity j

Figure 3. Proposed project schedule representation.



Activity list	1	2	4	3	5	6	9	8	7	10	11
Assigned resources list	E1, E3	E2, E3	E3	E2	E3	E4, E5	E3	E2	E3	E2, E5	E1, E2
	1	2	3	4	5	6	7	8	9	10	11

The positions of the assigned resources list refer to the activities of project

Figure 4. Feasible encoded solution to the project example.

4.1.1. *Decoding of solutions*

The serial method was considered in order to build the schedule related to the representation (Kolisch and Hartmann 1999). This method schedules the activities, one by one, in the order given by the activity list (forward scheduling). In Figure 3(a), activity j represents the ith activity that must be inserted in the schedule. When activity j is chosen to be inserted in the schedule, all its predecessors, which appear in some position $[1, (i-1)]$ in the activity list, will have already been scheduled. Then, activity j obtains the earliest feasible starting time. Specifically, activity j can start its development after all its predecessors have been completed, and when all the employees assigned to the activity are available (in Figure 3(b), position j details the employees, of every skill k, assigned to activity j). Thus, the related schedule is always a feasible one.

It is important to note that when the serial method is applied, one and only one schedule can be deduced from a given encoded solution, but different encoded solutions could be transformed in the same schedule (Kolisch and Hartmann 1999).

Figure 4 shows one feasible encoded solution to the project example described in Section 2.1. Figure 2 (Section 2.1) presents the schedule built from this solution by the serial method. The schedule details the specific employees assigned to each activity and the starting time defined for each activity.

4.2. ***Initial population***

The initial population contains a specific number of feasible solutions to the project to be scheduled (*i.e.* each solution of this population represents a feasible schedule to the project). Each solution of the initial population is randomly generated. Using a random approach guarantees a good level of genetic diversity in the initial population (Goldberg 2007). Such diversity is meant to prevent the premature convergence of the algorithm.

In order to obtain each solution, a two-stage process was used. The first stage determines the positions of the project activities on the activity list. The second stage defines the employees that are assigned to each activity on the basis of the human resource requirements for those activities (*i.e.* the second stage defines the assigned resources list).

The first stage is developed as follows. It begins with an empty activity list. The next activity for the list is randomly taken from the activities not yet inserted on the list while all its predecessors have already been inserted in it. Thus, each activity on the list is presented in a random position following all its predecessors.

The second stage assigns employees to each project activity. Each activity j requires r_{jk} employees with skill k. For each activity j, r_{jk} employees with skill k are randomly selected from the group of available employees with skill k, AR_k, and the selected employees are assigned to activity j (*i.e.* are assigned to position j of the assigned resources list).

This random process guarantees that the precedence relationships between the activities are not violated and that the human resource requirements for each activity are met during the construction of each solution.

4.3. *Fitness function*

The fitness function evaluates a given solution in relation to each one of the predefined optimization objectives, and then it defines a scalar fitness value for the solution on the basis of the results obtained by the evaluations. In this case, two objectives are considered. One objective is to maximize the effectiveness level of the sets of employees assigned to the project activities, while the other is to minimize the project makespan.

Given a solution to a project p, the fitness function decodes the schedule s related to the solution by using the method described in Section 4.1.1. Then, the function calculates the value of the term $e(s)$ corresponding to s (Equations 1 and 2), and calculates the value of the term $d(s)$ corresponding to s (Equations 3 and 4).

To calculate the term $e(s)$, the function utilizes the values of the terms $e_{rCr,j,k,g}$ inherent to s (Equation 2). As was mentioned in Section 2, the values of the terms $e_{rCr,j,k,g}$ inherent to each available employee r are known. The term $e(s)$ takes a real value over $[0, \ldots, N]$.

To calculate the term $d(s)$, the fitness function considers the values of the terms $ft(j, s)$ inherent to s (Equation 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.

In the literature, different approaches define a scalar fitness value on the basis of a set of values corresponding to different optimization objectives (Eiben and Smith 2007, Goldberg 2007, Coello Coello *et al.* 2007, Deb 2009). Some of these approaches define 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. For instance, one of the most applied approaches assigns to each solution a fitness value equal to the solution's dominance grade (Konak *et al.* 2006, Coello Coello *et al.* 2007, Deb 2009). This approach has been effectively used in previous works that propose multi-objective evolutionary algorithms to solve multi-objective project scheduling problems (Hanne and Nickel 2005). It also has been effectively used in previous works that propose multi-objective evolutionary algorithms to solve other kinds of multi-objective scheduling problem (Lau *et al.* 2009). In this article, the dominance grade is considered as the criterion to define a scalar fitness value for each solution.

The dominance grade of a solution is defined by the number of solutions that it dominates. Formally, the dominance grade of a solution x is defined by Equation (7). In this equation, the term P represents the population on which the dominance grade is calculated.

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

The value of the term P (Equation 7) depends on the process that calls the fitness function. The evolutionary algorithm has two processes that call the fitness function and use the values returned by this function. One of the processes is the selection of parents. The other process is the selection developed by the deterministic crowding strategy to define a new population. The selection of parents is developed on the current population. Thus, in this case, the term P is equal to the current population. However, the selection inherent to the deterministic crowding strategy is developed on the set composed of the solutions of the current population as well as the new solutions generated from it. Therefore, in this case, the term P is equal to the set composed of the solutions of the current population and the new solutions generated from it.

4.4. *Selection of parents*

This process selects a number of solutions from among the current population to conform pairs of solutions (parent solutions), which will be used to generate new solutions (offspring solutions) by the crossover and mutation processes. When selecting a solution, this process considers the fitness values of the solutions and is usually biased by a random factor. Thus, the solutions with the highest fitness values will have more chances of being selected.

There are several schemes to develop the selection process. The 2-tournament selection scheme, one of the most applied in the literature (Eiben and Smith 2007, Goldberg 2007), was applied. In this scheme, two solutions are randomly selected from the current population and compete for survival. The better one (*i.e.* the solution with the highest fitness value) is selected and becomes the first member of the pair. The other one is returned to the population. This operation is repeated to obtain the second member of the pair. The whole process is repeated until a number $M/2$ of pairs is obtained, where M is the population size.

4.5. *Crossover*

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

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

The crossover operators are directly applied to pairs of encoded solutions. Thus, the crossover must be designed on the basis of the representation defined for the solutions. In this case, the proposed representation consists of two lists: an activity list and an assigned resources list. Therefore, a crossover operator containing a feasible crossover operation for each of the mentioned lists is proposed.

4.5.1. *Crossover operation for activity lists*

This operation is applied to the activity lists of two parent solutions that must be recombined. The result of this operation consists of two new activity lists. The first one is assigned to the first offspring and the second one is assigned to the second offspring.

This operation is described below. First, the operation defines a random crossover point c, considering c between 1 and N. Then, the first c activities on the list of parent 1 are positioned in the first c positions on the list of offspring 1, in the same order. Subsequently, the operation copies the activities not included in the list of offspring 1 in the empty positions of this list. The activities not included in the list of offspring 1 are copied taking into account the order in which they appear in the list of parent 2. In this way, 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.

Figure 5 shows an example of the crossover operation described above for activity lists. In this example, the operation is applied to the activity lists corresponding to two parent solutions defined for the project example presented in Section 2.1.

This crossover operation corresponds to the one-point crossover developed by Hartmann (1998).

4.5.2. *Crossover operation for assigned resources lists*

This operation is applied to the assigned resources lists of two parent solutions that must be recombined. 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.

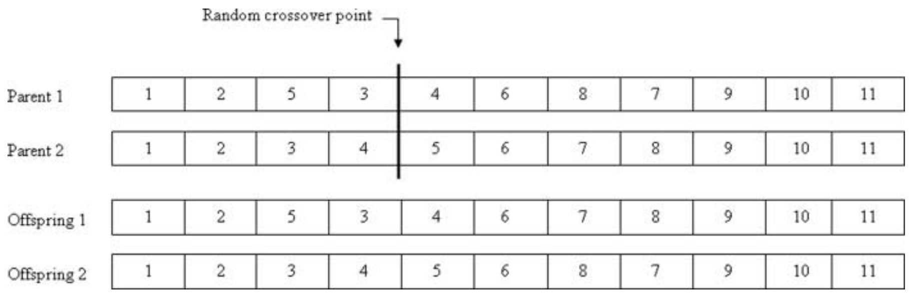

Random crossover point
Parent 1: 1 2 5 3 4 6 8 7 9 10 11
Parent 2: 1 2 3 4 5 6 7 8 9 10 11
Offspring 1: 1 2 5 3 4 6 7 8 9 10 11
Offspring 2: 1 2 3 4 5 6 8 7 9 10 11

Figure 5. Example of the crossover operation considered for activity lists.

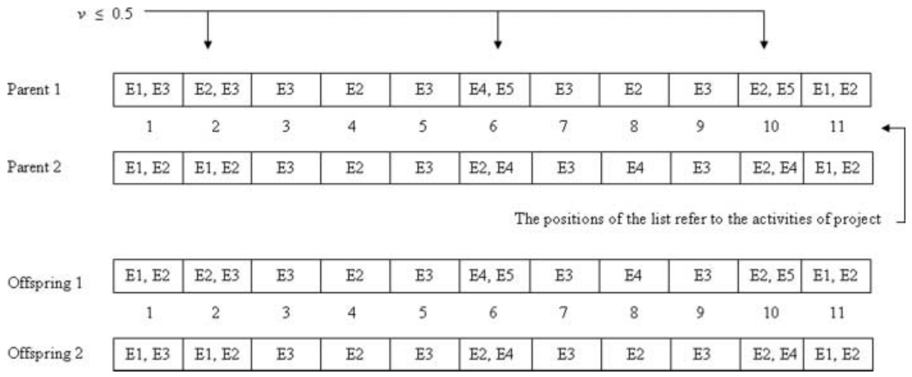

v ≤ 0.5
Parent 1: E1, E3 | E2, E3 | E3 | E2 | E3 | E4, E5 | E3 | E2 | E3 | E2, E5 | E1, E2
1 2 3 4 5 6 7 8 9 10 11
Parent 2: E1, E2 | E1, E2 | E3 | E2 | E3 | E2, E4 | E3 | E4 | E3 | E2, E4 | E1, E2
The positions of the list refer to the activities of project
Offspring 1: E1, E2 | E2, E3 | E3 | E2 | E3 | E4, E5 | E3 | E4 | E3 | E2, E5 | E1, E2
1 2 3 4 5 6 7 8 9 10 11
Offspring 2: E1, E3 | E1, E2 | E3 | E2 | E3 | E2, E4 | E3 | E2 | E3 | E2, E4 | E1, E2

Figure 6. Example of the crossover operation considered for assigned resources lists.

For each activity j of the project, the operation chooses a random value v over [0, 1], if $v \leq 0.5$, the resource assignment for j 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 $v > 0.5$, the resource assignment for j 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 lists.

Figure 6 shows an example of the crossover operation described above for assigned resources lists. In this example, the operation is applied to the assigned resources lists corresponding to two parent solutions defined for the project example presented in Section 2.1.

4.6. *Mutation*

This process randomly alters one or more characteristics of some solutions obtained after the crossover process. The mutation operator aims at introducing genetic diversity in the population (Goldberg 2007). The mutation operators are directly applied to encoded solutions. Thus, the mutation must be designed on the basis of the representation defined for the solutions. Therefore, a mutation operator that contains a feasible mutation operation for activity lists and a feasible mutation operation for assigned resources lists is proposed.

4.6.1. *Mutation operation for activity lists*

This operation is applied to the activity list of a given solution. The result of this operation consists of a new activity list for the solution. For each activity on the activity list, a new position is randomly chosen. In particular, the new position must be higher than the position corresponding

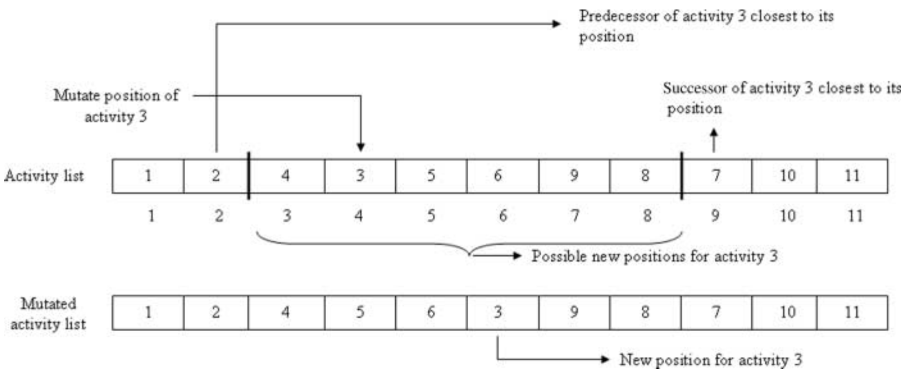

Predecessor of activity 3 closest to its position
Mutate position of activity 3
Successor of activity 3 closest to its position
Activity list
1 2 4 3 5 6 9 8 7 10 11
1 2 3 4 5 6 7 8 9 10 11
Possible new positions for activity 3
Mutated activity list
1 2 4 5 6 3 9 8 7 10 11
New position for activity 3

Figure 7. Example of the mutation operation considered for activity lists.

to any of the activity's predecessors, and lower than the position corresponding to any of the activity's successors. The activity is inserted in the new position with a probability of P_m. By this procedure, only precedence feasible lists are generated.

Figure 7 shows an example of this mutation operation. In this example, the operation is applied to the activity list corresponding to a solution defined for the project example presented in Section 2.1.

This mutation operation is an adaptation of the procedure proposed by Boctor (1996) in his simulated annealing algorithm to generate a neighbour.

4.6.2. *Mutation operation for assigned resources lists*

This operation is applied to the assigned resources list of a given solution. The result of this operation consists of a new assigned resources list for the solution. For each activity of the project, the operator defines a new resource assignment with a probability of P_m. In case a new resource assignment needs to be defined for a given activity, the operator considers the second stage of the mechanism used to create the random solutions of the initial population (Section 4.2). This operation always leads to new feasible lists.

Figure 8 shows an example of this mutation operation. In this example, the operation is applied to the assigned resources list corresponding to a solution defined for the project example presented in Section 2.1.

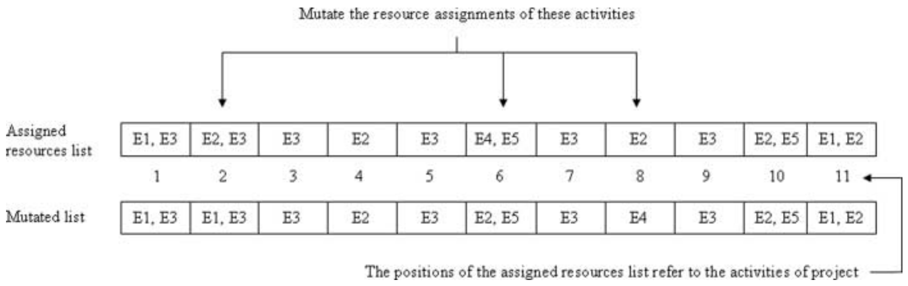

Mutate the resource assignments of these activities
Assigned resources list
E1, E3 E2, E3 E3 E2 E3 E4, E5 E3 E2 E3 E2, E5 E1, E2
1 2 3 4 5 6 7 8 9 10 11
Mutated list
E1, E3 E1, E3 E3 E2 E3 E2, E5 E3 E4 E3 E2, E5 E1, E2
The positions of the assigned resources list refer to the activities of project

Figure 8. Example of the mutation operation considered for assigned resources lists.

4.7. ***Replacement strategy***

This strategy is used to create a new population from the solutions in the current population (parent solutions) and the new generated solutions (offspring solutions).

Different replacement strategies have been proposed in the literature (Eiben and Smith 2007). The strategy called deterministic crowding has been applied (Eiben and Smith 2007). This strategy preserves the best solutions obtained and maintains the diversity of the population.

In the deterministic crowding strategy, offspring compete directly with their respective parents. This strategy works as follows. Every offspring competes with one of its parents (*i.e.* with its closest parent) and may replace it if the offspring is not worse (*i.e.* if the offspring has a fitness value greater than or equal to the fitness value of the parent). The closeness between an offspring solution and a parent solution is defined based on the distance between their fitness values.

5. Computational experiments

In this section, the computational experiments developed to evaluate the performance of the proposed multi-objective evolutionary algorithm are described. Then, the results obtained are presented and analysed.

Test instances of the standard sets j30, j60 and j120 from PSPLIB (Kolisch *et al.* 1995, Kolisch and Sprecher 1997) were selected, and then the content of the selected instances was extended with the aim of adapting these instances to the characteristics of the addressed problem.

Table 6 shows the characteristics of the instance sets defined to develop the experiments. Sets j30_5, j30_10 and j30_15, j60_5, j60_10 and j60_15, and j120_5, j120_10 and j120_15 contain selected instances of j30, j60 and j120, respectively. Moreover, the defined sets have no instances in common.

Each instance selected from PSPLIB contains information about a precise number of activities. For each activity, the instance details the duration, the precedence relations and the number of required resources for each possible type of resources. Each instance also contains information about the available resources. Specifically, each instance defines four types of available resources, and a number of available resources for every type. Each type of resources is considered to be a different skill, and the number of available resources for each type to correspond to the number of available employees for each skill. Moreover, it is assumed that each employee masters only one of the four possible skills.

The content of the instances selected from PSPLIB was extended. Specifically, for each instance selected from PSPLIB, all the terms $e_{rCr,j,k,g}$ inherent to each employee r of the instance were defined - these terms are defined considering each of the possible work contexts for r in the instance - and a random value over [0, 1] for each of the above-mentioned terms was also defined.

Each selected instance from PSPLIB has a known optimal solution in respect to the project makespan. These optimal solutions are provided by PSPLIB. However, the instances selected from PSPLIB have no known optimal solutions in respect to the effectiveness level of the sets of

Table 6. Characteristics of instance sets.

Instance set	No. of activities to be planned in each instance	No. of possible sets of employees per activity	No. of instances
j30_5	30	1 to 5	40
j30_10	30	1 to 10	40
j30_15	30	1 to 15	40
j60_5	60	1 to 5	40
j60_10	60	1 to 10	40
j60_15	60	1 to 15	40
j120_5	120	1 to 5	40
j120_10	120	1 to 10	40
j120_15	120	1 to 15	40

Table 7. Parameter values of the evolutionary algorithm.

Parameter	Value
Crossover probability P_c	0.8
Mutation probability P_m	0.05
Population size	50
Number of generations	500

employees assigned to the project activities (*i.e.* the project effectiveness level). For that reason, for each extended instance, an optimal solution in respect to the project effectiveness level was designed with the aim of using it as a reference. Specifically, for each extended instance, a feasible solution s was designed. Solution s was designed by the mechanism used to create the solutions of the initial population of the evolutionary algorithm (Section 4.2). Then, all existing terms $e_{rCr,j,k,g}$ in the solution s were defined. A value equal to 1 (maximal value) was assumed for each of the terms, and they were added to the content of the instance, together with the assumed values. When adding the assumed values to the content of the instance, the project effectiveness level $e(s)$ of the solution s is equal to N (N is the number of activities of the instance). Thus, a feasible solution s with a level $e(s)$ equal to N for the instance was defined. Considering that N is the maximal level for the solutions of an instance with N activities, the defined solution s can be considered as an optimal solution in relation to the project effectiveness level.

Thus, each extended instance has a known optimal solution in respect to the project makespan (*i.e.* the optimal solution provided by PSPLIB) and a known optimal solution in respect of the project effectiveness level. These optimal solutions are considered here as references.

The proposed multi-objective evolutionary algorithm has been tested 20 times on each of the extended instances. After each one of the 20 runs, the algorithm provided the non-dominated solution set of the last population or generation. Table 7 gives the parameter values used for these experiments. The parameters were fixed thanks to preliminary experiments that showed that these values led to the best and most stable results.

To evaluate the performance of the multi-objective evolutionary algorithm, some aspects of the quality of the non-dominated solution sets obtained by the experiments were analysed and the computation times required by the algorithm were also analysed.

First, the spread and distribution of the solutions in the obtained sets were analysed. To analyse the two mentioned aspects, the distribution and spread metrics proposed by Deb (2009) were utilized. Thus, it was found that in the obtained sets, the solutions are well distributed and widely spread (*i.e.* for each optimization objective, a wide and varied range of values is covered by the non-dominated solutions). This means that the obtained sets contain solutions with good values for the project makespan and poor values for the project effectiveness level, solutions with poor values for the project makespan and good values for the project effectiveness level, and solutions with intermediate values for each optimization objective.

The accuracy of the solutions in the sets obtained by the experiments was also analysed. To analyse this aspect, the accuracy metric utilized in Hanne and Nickel (2005) was applied. This metric does not require knowledge of the true non-dominated solution sets of the instances, which are unknown in the present case. The metric only needs a reference value for each optimization objective, and in this case, each instance has a known optimal value for each objective. Considering the 20 sets obtained by the algorithm for each instance, the metric selects the non-dominated solution with the minimal project makespan of each set, and calculates the average percentage deviation of these solutions from the optimal project makespan. Note that each instance has a known optimal solution with respect to the project makespan. Then, the metric selects the non-dominated solution with the maximal project effectiveness level of each set, and calculates the

Table 8. Average percentage deviation obtained for each instance set in respect of each optimization objective, and average computation time obtained for each instance set.

Instance set	*max.* (*e*)	*min.* (*d*)	Average time (sec)
j30_5	0	0.02	10.56
j30_10	0	0.012	13.2
j30_15	0.2	0.009	15.84
j60_5	0	0.17	34.08
j60_10	0.4	0.09	39.76
j60_15	1.8	0.07	45.44
j120_5	1.1	0.25	124.56
j120_10	1.25	0.19	152.24
j120_15	2.7	0.18	193.76

average percentage deviation of these solutions from the optimal project effectiveness level. Note that each instance has a known optimal solution with respect to the project effectiveness level. Finally, for each set of instances, the metric calculates the average value of the average percentage deviations for each optimization objective.

Table 8 reports the average percentage deviation *Av. Dev.* (%) obtained for each instance set in respect of each optimization objective. Column 2 presents the *Av. Dev.* (%) obtained for each instance set in respect of the maximization of the project effectiveness level, and column 3 presents the *Av. Dev.* (%) obtained for each instance set in respect of the minimization of the project makespan. Table 8 also 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 with 3 GB RAM under Windows XP Professional Version 2002.

Regarding the maximization of the project effectiveness level, the *Av. Dev.* (%) obtained by the evolutionary algorithm for j30_5, j30_10 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 evolutionary algorithm for j30_15, j60_10, j60_15, j120_5, j120_10 and j120_15 is greater than 0%. Considering that the instances of j30_15, 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 30, 60 and 120, respectively, the meaning of the average deviation obtained for each one of these sets was analysed. In the case of j30_15, an average deviation equal to 0.2% indicates 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 29.94. In the case of j60_10 and j60_15, average deviations equal to 0.4% and 1.8% indicate that the average level of the non-dominated solutions selected from the obtained sets is 59.76 and 58.92 respectively. In the case of j120_5, j120_10 and j120_15, average deviations equal to 1.1%, 1.25% and 2.7% indicate that the average level of the non-dominated solutions selected from the obtained sets is 118.68, 118.5 and 116.76, respectively. Based on these, it may be stated that the algorithm has obtained non-dominated solutions with very high project effectiveness levels for the instances of j30_15, j60_10, j60_15, j120_5, j120_10 and j120_15.

As regards the minimization of the project makespan, the *Av. Dev.* (%) values obtained for the instance sets are lower than 0.26%. These results indicate that, for each instance, the average project makespan of the non-dominated solutions selected from the sets obtained by the algorithm (*i.e.* solutions with the minimal project makespan) is near to the optimal project makespan. Therefore, it may be stated that the algorithm has obtained non-dominated solutions with near-optimal makespans for the instances of each set.

From the previous analysis of the average deviations presented in Table 8, the following may be pointed out. 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 j30_15, 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.

Regarding the average computation times presented in Table 8, the following points may be mentioned. For j30_5, j30_10 and j30_15, the average time required by the algorithm was lower than 16 seconds. For j60_5, j60_10 and j60_15, the average time required by the algorithm was higher than 34 seconds and lower than 46 seconds. Then, for j120_5, j120_10 and j120_15, the average time required by the algorithm was higher than 124 seconds and lower than 194 seconds. Taking into consideration the complexity of the instance sets used, particularly the complexity of the instances of j120_5, j120_10 and j120_15, it is considered that the average computation times obtained by the algorithm for the nine instance sets are acceptable.

6. Conclusions

The problem of scheduling a project has been addressed with the aim of assisting project managers at the early stage of designing project schedules. Thus, as part of the problem, two priority optimization objectives were considered for managers at that stage. One objective is to minimize the project makespan. The other objective is to maximize the project effectiveness level (*i.e.* to maximize the effectiveness level of the sets of employees assigned to the project activities). In relation to the second objective, it was considered that the effectiveness of a set of employees depends on the effectiveness of each one of the employees belonging to the set. On the other hand, it was considered that the effectiveness level of an employee depends on his or her work context. Therefore, it is possible to define different effectiveness levels in relation to different work contexts for each employee. This is a really significant aspect because the effectiveness of an employee in a real project depends on many contextual factors (Heerkens 2002, Wysocki 2003). Such factors involve the attributes of the activity to which the employee is assigned, the other employees with whom the employee in question must work, and the experiences and the attributes of the employee (Barrick *et al.* 1998, Heerkens 2002). Therefore, the effectiveness of an employee needs to be considered in relation to his or her work context. To the best of the authors' knowledge, the influence of the work context on the effectiveness of the employees has not been considered in previous related works.

To solve the problem, a multi-objective evolutionary algorithm has been proposed. Considering a given project, this algorithm designs feasible schedules for the project, and evaluates the designed schedules in relation to each one of the defined optimization objectives. In particular, regarding the maximization of the project effectiveness level, the evaluation is developed on the basis of knowledge about the effectiveness of employees involved in each schedule. Specifically, for each designed schedule, the algorithm estimates the effectiveness level of the sets of employees assigned to the project activities. The effectiveness of a set assigned to a particular activity is estimated on the basis of the effectiveness level of the employees included in the set. The evolutionary algorithm provides an approximation to the Pareto set (*i.e.* a number of near non-dominated solutions) as a solution to the problem addressed here.

Computational experiments were developed to evaluate the performance of the proposed algorithm. The experiments consisted in solving instances corresponding to nine different sets:

j30_5, j30_10, j30_15, j60_5, j60_10, j60_15, j120_5, j120_10 and j120_15. The instances belonging to different sets have different complexity (*i.e.* have a different search space). Thus, the utilization of those sets has made it possible to evaluate the evolutionary algorithm performance over instances with a different degree of complexity.

In the experiments, the algorithm was tested many times on each of the instances, 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 algorithm on each instance, different aspects of the quality of the obtained sets of non-dominated solutions were analysed. Specifically, the spread, distribution and accuracy of the solutions in the obtained sets were analysed.

On the basis of the above-mentioned analysis, it may be stated that the algorithm has obtained wide, well-distributed sets of non-dominated solutions. In addition, in the obtained sets, the solution with the minimal project makespan of the set has a near-optimal makespan, and the solution with the maximal project effectiveness level of the set has a very high or an optimal project effectiveness level. Thus, the non-dominated solution sets obtained by the algorithm provide a number of solutions with different trade-offs between the optimization objectives. Based on the previously mentioned aspects, it is considered that a project manager could obtain a number of schedules with different trade-offs between the optimization objectives by using the algorithm. Then, he or she should choose the solution(s) that closely match his or her preferences about the trade-off between the optimization objectives.

In future works, other relevant optimization objectives in the addressed problem will be included (*e.g.* the minimization of the project cost). New feasible crossover and mutation processes will be proposed for the defined representation of solutions. In addition, different approaches that have been proposed in the literature will be evaluated to define a scalar fitness value for each solution of a multi-objective problem.

Considering that the proposed algorithm uses knowledge about the effectiveness of the employees, future research should be conducted in order to develop approaches that automatically estimate the effectiveness levels of the employees from available information about the participation of the employees in already executed projects. These approaches will complement this algorithm.

References

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

Barrick, M.R., *et al.*, 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*: *Proceedings of the 7th international conference on the practice and theory of automated timetabling (PATAT 2008)*, 18–22 August, Montreal. Montreal: Université de Montréal, 1–4.

Bellenguez, O. and Néron, E., 2004. Methods for solving the multi-skill project scheduling problem. *In*: *Proceedings of the 9th international workshop on project management and scheduling (PMS'2004)*, 26–28 April, Nancy, France. Nancy, France: Université Nancy, 66–69.

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

Bellenguez, O. and Néron, E., 2007. A branch-and-bound method for solving multi-skill project scheduling problem. *RAIRO—Operations Research*, 41 (2), 155–170.

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

Boctor, F.F., 1996. Resource-constrained project scheduling by simulated annealing. *International Journal of Production Research*, 34 (8), 2335–2351.

Chan, F.T.S., Wong, T.C., and Chan, L.Y., 2006. Flexible job-shop scheduling problem under resource constraints. *International Journal of Production Research*, 44 (11), 2071–2089.

Coello Coello, C.A., Lamont, G.B., and Veldhuizen, D.A., 2007. *Evolutionary algorithms for solving multi-objectives problems*. 2nd ed. New York: Springer.

Cortés Rivera, D., Landa Becerra, R., and Coello Coello, C.A., 2007. Cultural algorithms, an alternative heuristic to solve the job shop scheduling problem. *Engineering Optimization*, 39 (1), 69–85.

Deb, K., 2009. *Multi-objective optimization using evolutionary algorithms*. New York: Wiley.

Drezet, L.E. and 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. and Smith, J.E., 2007. *Introduction to evolutionary computing*. 2nd ed. Berlin: Springer.

Focacci, F., Laborie, P., and Nuijten, W., 2000. Solving scheduling problems with setup times and alternative resources. *In*: *Proceedings of the fifth international conference on artificial intelligence planning and scheduling (AIPS 2000)*, 14–17 April, Breckenridge, CO. Palo Alto, CA: AAAI, 92–111.

Goldberg, D.E., 2007. *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley.

Gutjahr, W.J., *et al.*, 2008. Competence-driven project portfolio selection, scheduling and staff assignment. *Central European Journal of Operations Research*, 16 (3), 281–306.

Hanne, T. and 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*. New York: McGraw-Hill.

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

Kolisch, R. and 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*. New York: Kluwer Academic, 147–178.

Kolisch, R. and Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: an update. *European Journal of Operational Research*, 174 (1), 23–37.

Kolisch, R. and Sprecher, A., 1997. PSPLIB—a project scheduling library. *European Journal of Operational Research*, 96 (1), 205–216.

Kolisch, R., Sprecher, A., and Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41 (10), 1693–1703.

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

Lau, H.C.W., *et al.*, 2009. A fuzzy guided multi-objective evolutionary algorithm model for solving transportation problem. *Expert Systems with Applications*, 36 (4), 8255–8268.

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

Néron, E., 2002. Lower bounds for the multi-skill project scheduling problem. *In*: *Proceedings of the eighth international workshop on project management and scheduling (PMS'2002)*, 3–5 April, Valencia, Spain. Valencia, Spain: University of Valencia, 274–277.

Néron, E., Bellenguez, O., and Heurtebise, M., 2006. Decomposition method for solving multi-skill project scheduling problem. *In*: *Proceedings of the tenth international workshop on project management and scheduling (PMS'2006)*, 26–28 April, Poznañ. Poznañ: Wydawnictwo Nakom, 265–269.

Valls, V., *et al.*, 2007. Project scheduling optimization in service centre management. *Tijdschrift voor Economie en Management*, 52 (3), 341–366.

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

Wong, T.C., Chan, F.T.S., and Chan, L.Y., 2009. A resource-constrained assembly job shop scheduling problem with lot streaming technique. *Computers and Industrial Engineering*, 57 (3), 983–995.

Wysocki, R. K., 2003. *Effective project management*. 3rd ed. New York: Wiley.