Journal of Manufacturing Systems 63 (2022) 506–523

Contents lists available at ScienceDirect

Journal of Manufacturing Systems

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

Bi-objective resource-constrained project scheduling problem with time-dependent resource costs

Javier Alcaraz [a,b], Laura Anton-Sanchez [a,b,*], Francisco Saldanha-da-Gama [c,d]

[a] Departamento de Estadística, Matemáticas e Informática, Universidad Miguel Hernández, 03202 Elche, Alicante, Spain
[b] Centro de Investigación Operativa, Universidad Miguel Hernández, 03202 Elche, Alicante, Spain
[c] Departamento de Estatística e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal
[d] Centro de Matemática, Aplicações Fundamentais e Investigação Operacional, Faculdade de Ciências, Universidade de Lisboa, 1749-016 Lisboa, Portugal

ARTICLE INFO

Keywords:
Resource-constrained project scheduling problem
Time-dependent resource costs
Makespan
Multi-objective optimization
Exact Pareto front
Metaheuristics

ABSTRACT

This work provides new insights on bi-criteria resource-constrained project scheduling problems. We define a realistic problem where the objectives to combine are the makespan and the total cost for resource usage. Time-dependent costs are assumed for the resources, i.e., they depend on when a resource is used. An optimization model is presented and it is followed by the development of an algorithm aiming at finding the set of Pareto solutions. The intractability of the optimization models underlying the problem also justifies the development of a metaheuristic for approximating the same front. We design a bi-objective evolutionary algorithm that includes problem-specific knowledge and is based on the Non-dominated Sorting Genetic Algorithm (NSGA-II). The results of extensive computational experiments performed using instances built from those available in the literature are reported. The results demonstrate the efficiency of the metaheuristic proposed.

1. Introduction

The resource-constrained project scheduling problem (RCPSP) consists of scheduling a set of activities subject to precedence and resource constraints. This is a well-known problem with much work available in the literature. This is attested by the large number of surveys that have been published. Among these, we quote the most recent ones by Abdolshah [2], Habibi et al. [23] and Hartmann and Briskorn [27,28]. The amount of work done on the RCPSP has reached such a volume that one even finds surveys on specific types of methodologies developed for it. For instance, Pellerin et al. [42] focus on hybrid metaheuristics in the context of these problems.

This work focuses on a bi-criteria RCPSP considering the makespan and the total costs for resource usage as the objectives to optimize. We consider resource- and time-dependent costs, i.e., the cost depends on the resource being considered as well as on the time it is used. We can find many different examples in practice of such time-dependent resource costs. This holds, for instance, when scarce resources are involved, such as water in some geographies. The availability of such resource may easily change over time (e.g. summer versus autumn) with impacts on its costs. In the specific context of manufacturing systems, this issue becomes of great relevance with energy costs that are typically much cheaper during off-peak times than in peak times (an interesting discussion is presented by Moon and Park [39]). Additionally, the labor costs may easily depend on the time the resources are used (e.g. weekdays versus weekends). What is more, all these costs may be affected by an annual increase induced by the consumer price index. Above all, by capturing more realistic settings, it is possible to develop more adequate tools of great relevance in the context of smart manufacturing in general [46] and in the context of scheduling activities in manufacturing systems in particular [37].

The goal for the bi-objective problem we investigate in this work is to find the Pareto front, i.e., the entire set of solutions that cannot be improved in terms of one objective without deteriorating the other. The intractability of the optimization models required to find those solutions, that will be clear by the experiments we report, motivates the development of a heuristic algorithm for approximating the front. We propose a metaheuristic based on the Non-dominated Sorting Genetic Algorithm (NSGA-II) proposed by Deb et al. [18].

We start by discussing modeling issues related with the problem. Afterwards, we discuss an exact algorithm for finding exact Pareto solutions, namely, the so-called AUGMECON [36] that we adapt to our problem. The work proceeds with the new metaheuristic we propose for approximating the Pareto front. Finally, our methodological

* Corresponding author at:Departamento de Estadística, Matemáticas e Informática, Universidad Miguel Hernández, 03202 Elche, Alicante, Spain.
E-mail address: l.anton@umh.es (L. Anton-Sanchez).

https://doi.org/10.1016/j.jmsy.2022.05.002
Received 7 December 2021; Received in revised form 8 April 2022; Accepted 2 May 2022
Available online 18 May 2022

contribution is assessed via a series of computational tests performed using instances built from those available in the literature. The results obtained are reported in detail.

The remainder of this paper is organized as follows. In Section 2 we discuss the relation between our work and the existing literature. Particular emphasis is put on multicriteria decision making in the context of the RCPSP as well as on the search for a time-cost trade-off. In Section 3, we provide all details of the problem we are studying and present a bi-criteria vector optimization model. We also illustrate the relevance of considering such model. In Section 4, we apply an exact algorithm for finding exact Pareto solutions. Section 5 presents the metaheuristic we have designed and developed for the problem. In Section 6, we report on the extensive experiments performed to assess the methodologies proposed. Finally, the paper ends with an overview of the work carried out.

2. Relation with the existing literature

The major aspects related to the current paper include multicriteria optimization in the context of the RCPSP as well as a cost-dependent resource usage. Next, we provide an overview of the work done, which in turn, helps to strengthen the motivation for developments we propose in the following sections.

2.1. Multicriteria optimization in the context of the RCPSP

The RCPSP is prone to the consideration of different objectives depending on the specific interests of the decision maker. We refer to [11] and [23] for many examples of different types of objectives of practical relevance. These include the makespan, the maximum lateness (when deadlines are considered), and cost objectives. Interestingly, the literature on project scheduling and management is not abundant when it comes to considering cost as a sole objective. Nevertheless, we still find some good exceptions such as the work by Martins [35], where the author discusses aspects such as the integration of cash-flows along with the project execution, or borrowing strategies for supporting projects' costs.

More often than not, more than one objective emerges as relevant in RCPSP. This justifies that large stream of research one can find in terms of multicriteria models and techniques for these problems. We note the relevance of this issue in [52], where the authors study a bi-objective version of the RCPSP such that, in addition to makespan minimization, it seeks to minimize the total tardiness (deadlines are assumed for the activities). Pareto solutions are sought by means of a two-stage algorithm: first, all the supported solutions are identified and the solution space is reduced considering the triangle areas where non-supported solutions can be found; afterwards, the non-supported solutions are identified. In this work, the authors aim to find exact Pareto solutions. Unfortunately, this can be accomplished for rather small instances, which explains the existence of a larger number of articles focusing on approximate procedures for multicriteria RCPSP as we can observe next.

Al-Fawzan and Haouari [5] consider a bi-objective RCPSP that in addition to the usual makespan objective, involves a robustness objective and they propose a Tabu Search-based algorithm for approximating the Pareto front. For every activity, the authors define a slack, representing the amount of time the activity can be shifted without delaying the start of their direct successors in the precedence network, while maintaining resource usage feasibility. The robustness measure adopted is the sum of such slacks for all activities because it somehow quantifies the ability of a schedule to cope with non-predictable changes in some activity(ies). A Tabu Search-based algorithm is developed for approximating the Pareto front. The same problem was treated by Abbasi et al. [1] who proposed a simulated annealing-based procedure.

Abello and Michalewicz [3] seek to minimize the makespan and the project cost (resource allocation) in a project scheduling problem with a time-dependent number of activities. The authors develop a multiobjective evolutionary algorithm for finding non-dominated solutions. Wang et al. [51] also investigate a bi-objective RCPSP. The authors assume that the processing time of the activities may change which will call for resource transfer decisions. One objective function represents the so-called total starting time criticality of the activities. For each activity, this value is computed as the product of a marginal cost for starting the activity later than initially planned and the probability that such activity cannot start according to its (initially) scheduled starting time. The second objective function represents the total resource transfer cost. For the above problem, the authors developed a non-dominated sorting genetic algorithm type II.

Wang et al. [53,54] study a three-objective RCPSP: they consider the two objective functions already adopted in [52] and add a third one that measures the workload balance level to be maximized. In that work, genetic algorithms are hybridized with the Self Controlling Dominance [45] for finding approximate Pareto solutions. Habibi et al. [24] also consider a three-objective RCPSP with time-dependent resource requirements and capacities. In addition to the traditional makespan, the authors also consider maximizing the schedule robustness (considering a weighted sum of the activities' free slacks), and maximizing the discounted cost associated with the resources. A non-dominated sorting genetic algorithm and a multi-objective particle swarm optimization procedure are proposed for approximating the Pareto front.

The above literature shows that, in the context of the RCPSP, most of the research effort has been placed in developing algorithms for approximating the Pareto front. The aforementioned single paper that seeks to identify exact Pareto solutions does so without exploring state-of-the-art tools in the context of bi-criteria optimization. Moreover, cost is not considered in that specific article.

We note that research can be found that aims at finding exact Pareto fronts for the most natural extension of the RCPSP: the multi-mode RCPSP. This is the case with Florez et al. [22], who consider three objective functions: the makespan, the total labor and investment cost, and resource stability. The authors assume that human resources are involved and thus some social objective is needed to better support decision making. Pareto solutions are found using an a priori lexicographic ordering of the objectives, followed by the application of the ε-constraint method.

2.2. Time-dependent costs

The vast existing literature on project management problems is abundant when it comes to searching for time-cost trade-off solutions. This was exactly the case when such problems started being investigated in the specialized literature stemming from the leading work by Kelley and Walker [30]. This seminal article set the idea of compressing the execution time of the activities with some cost incurred. Achuthan and Hardjawidjaja [4] proposed an interesting innovation to that idea: the cost of the activities is time-dependent. As the authors point out, this is quite realistic in many problems. Nevertheless, in that work, resources are abundant. Other work aligned with using time-dependent costs is that by Szmerekovsky and Venkateshan [48] in which, again, activity duration can be compressed although, once more, unlimited resources are assumed.

To the best of the authors' knowledge, in the context of the RCPSP, no work has explicitly considered time-dependent resource costs, which is a major feature in our current paper. Nevertheless, some works can be found capturing time-dependent features which, in some way, may reflect the existence of resources with time-dependent costs. Möhring et al. [38] mention the existence of time-dependent resource profiles in RCPSP (related with resource availability) although no costs are explicitly captured. Pottel and Goel [43] consider a RCPSP with time-dependent activity processing times and resource consumption. Again, costs are not explicitly considered. In the context of scheduling problems in manufacturing systems, Wu et al. [55] consider a multi-scenario setting, capturing uncertainty in the processing time of

the activities. The difference with respect to the two previous works is that the processing time is exogenous (although uncertain).

Overall, by reviewing the existing literature, we realize that the time-dependent cost associated to resources has never been considered for the RCPSP, let alone combined with the minimization of the makespan in a multicriteria framework, as we propose in this work.

2.3. Contribution provided by the current work

Given the existing literature as well as the modeling aspects and algorithmic developments proposed in the current paper, we can summarize the major contributions we provide as follows:

- The RCPSP is extended to capture time-dependent resource costs.
- The above extension leads to a cost-minimization objective that is considered together with the makespan minimization within a bi-criteria modeling framework.
- An exact procedure is developed for determining the exact Pareto front.
- A metaheuristic based on the NSGA-II algorithm is proposed for approximating the Pareto front aiming at tackling large-scale instances of the problem.
- We report on the results of an extensive set of computations performed by considering a well-known dataset of instances.

3. Problem details

A project consists of a set $V = \{0, 1, \ldots, n, n+1\}$ of activities such that for each activity $j \in V$, a duration or processing time d_j is known. The latter is assumed to be deterministic and integer. There are precedence relations between activities, all of a finish-start type. For each activity j, a set P_j is considered that contains all its predecessors. Preemption is not allowed, i.e., activities are executed during d_j time periods from their start time without interruption. Moreover, activities make use of a set K of renewable resources with each activity $j \in V$ requiring r_{jk} units of resource k per time unit. The availability of resource k in each time unit is given by B_k. The problem consists of finding the start time for each activity so that the precedence relations and the resource constraints are satisfied and the project makespan is minimized. Activities 0 and $n+1$ are dummy activities representing the start and the end of the project, respectively. These activities have a null processing time and do not consume resources.

Different models have been proposed for the RCPSP makespan minimization. A comprehensive overview is provided by Artigues et al. [9]. We take the first optimization model proposed for the model as a starting point, which is due to Pritsker et al. [44]. This model makes it necessary to set a planning horizon, of a certain length T. This indicates that all the activities must be completed by time T. Ideally, T should be determined by a sharp upper bound in the optimal makespan. Nonetheless, in the worst case scenario, it can be set equal to the sum of the duration of all activities. This induced planning horizon is thus a natural number, which is divided into unitary-time periods, starting at time 0.

Given the precedence-relations, for each activity $j \in V$ there is an earlier starting time, ES_j, which is the minimum time necessary to execute the direct and transitive predecessors of j. On the other hand, given that the project much be completed at time T (at most) then the starting time of an activity $j \in V$ should be such that it allows its direct transitive successors to be executed in a way that the entire project ends at time T or before. In other words, for every activity $j \in V$ there is a latest starting time, LS_j. These times, ES_j and LS_j, are calculated in the usual way that we revisit to make this document self-contained:

$$ES_0 = 0;$$
$$ES_j = \max_{i \in P_j}\{ES_i + d_i\} \quad \forall j \in V;$$

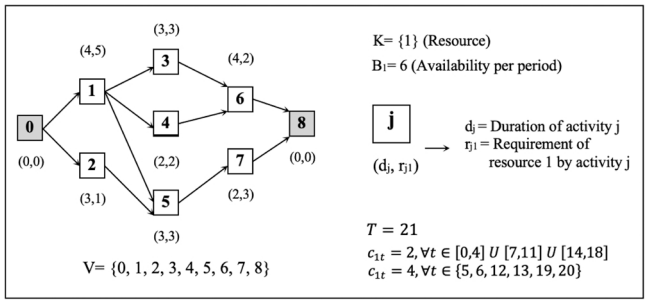


Fig. 1. An example of the RCPSP with time-dependent resource costs.

$$LS_{n+1} = T;$$
$$LS_j = \min_{k \,/\, j \in P_k}\{LS_k - d_j\} + T - ES_{n+1} \quad \forall j \in V.$$

Considering the above elements, the following decision variables are defined:

$$y_{jt} = \begin{cases} 1 & \text{if } j \text{ starts at time } t, \\ 0 & \text{otherwise,} \end{cases}$$

$\forall\, j \in V$ and $t \in \{ES_j, \ldots, LS_j\}$. For a vector of objective functions of interest, say $\mathbf{f(y)}$, a vector optimization RCPSP can be formulated as follows:

$$\text{minimize} \quad \mathbf{f}(\mathbf{y}) = (f_1(\mathbf{y}), f_2(\mathbf{y}), \ldots, f_L(\mathbf{y})), \tag{1}$$

$$\text{subject to} \quad \sum_{t=ES_j}^{LS_j} y_{jt} = 1 \quad \forall j \in V, \tag{2}$$

$$\sum_{t=ES_j}^{LS_j} t\, y_{jt} - \sum_{t=ES_i}^{LS_i} t\, y_{it} \geq d_i \quad \forall i, j \in V : i \in P_j, \tag{3}$$

$$\sum_{j \in V} r_{jk} \cdot \sum_{\tau=\max\{t-d_j+1, ES_j\}}^{\min\{t, LS_j\}} y_{jt} \leq B_k \quad \forall k \in K, t \in \{0, \ldots, T-1\}, \tag{4}$$

$$y_{jt} \in \{0, 1\} \quad \forall j \in V, t \in \{ES_j, \ldots, LS_j\}. \tag{5}$$

In this model, Constraints (2) ensure that every activity starts in some time; Constraints (3) are the precedence constraints and they guarantee that if some activity i precedes another activity j, then the latter can only start after the former has been finalized; the resource constraints are represented by (4), in each time period the amount available of each resource limits its usage. Finally, (5) define the domain of the y-variables.

What remains to be defined is the set of objective functions to consider. Using the y-variables, the makespan is straightforwardly defined as

$$f_1(\mathbf{y}) = \sum_{t=ES_{n+1}}^{T} t\, y_{(n+1),t} \tag{6}$$

Let us consider now that we have a time-dependent cost for the use of the renewable resources. In particular, we denote by c_{kt} the cost of employing one unit of resource k in period between times t and $t+1$, for all $k \in K$ and $t \in \{0, \ldots, T-1\}$.

The total cost for resource usage becomes

$$f_2(\mathbf{y}) = \sum_{j \in V \setminus \{n+1\}} \sum_{t=\max\{0, ES_j\}}^{\min\{T-1, LS_j\}} \left(y_{jt} \sum_{\tau=t}^{t+d_j-1} \sum_{k \in K} r_{jk}\, c_{k\tau} \right) \tag{7}$$

Example 1. We present an example for illustrating the above problem. Fig. 1 shows an instance of a project with time-dependent costs. The example has been adapted from that presented in [6]. The project consists of 7 activities making use of a single renewable resource, which has an availability of 6 units per period. Activities 0 and 8

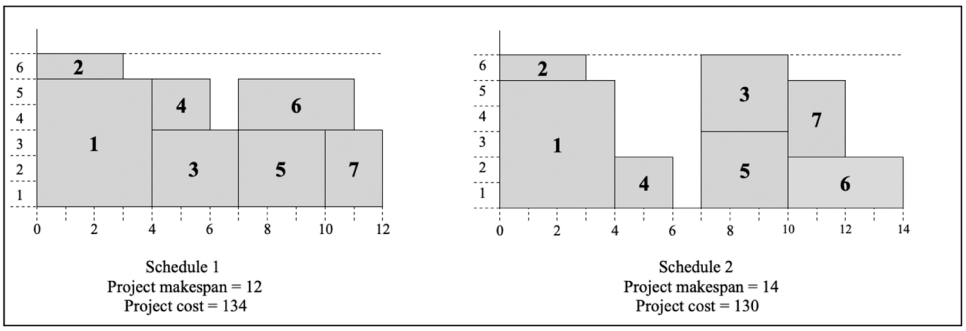


Fig. 2. Different solutions to project in Fig. 1.

represent the usual dummy activities, which have also been included in set V. The length of the planning horizon for this project was set equal to the sum of all the processing times. In each period, the cost per unit of the renewable resource, which presents two different rates, has been calculated.

Considering the objectives $f_1(\mathbf{y})$ and $f_2(\mathbf{y})$ separately, we obtain different solutions to the problem. Two different feasible solutions have been presented in Fig. 2, where the values for both objectives have been calculated. Both schedules represent a way to execute the same project, but, as the activities are executed in different periods, the cost may differ. Looking at both solutions, we can conclude that none of them dominates the other in the sense that no objective can be improved without deteriorating the other. As we will also discuss, this is in line with one of the aims of this work, which will be to find the so-called Pareto front, i.e., the set of non-dominated solutions.

In terms of the resource-constrained project scheduling problem, a major novelty of our work stems from considering the objective function (7). This fully changes the nature of the problem. First, the concept of optimality is no longer valid since we now have two objective functions gathered within a bi-criteria model. What is more, as our illustrative example reveals, the new objective function and the usual makespan (to minimize) can easily be conflicting. This poses a major challenge when it comes to solving the model and also selecting a solution. In particular, the best we can now do is to look for the so-called Pareto solutions, i.e., those solutions such that an objective function value cannot be improved without deteriorating the other. In the example, if our objective is only to minimize the project duration, the optimal makespan is 11. However, if we consider both objectives, there would be 7 Pareto solutions, which vary from a schedule with a makespan of 11 and a cost of 140 to a schedule with a makespan of 18 and a cost of 118. Between these two solutions it may be possible to provide a decision maker with a rich set of alternatives from which a better decision could certainly be made.

In the following sections we propose an algorithm for finding exact Pareto solutions and afterwards an approximate algorithm, since for large-scale instances finding exact Pareto solutions may turn out to be cumbersome.

4. Finding exact Pareto solutions

Following the previous section, we are focusing on problem

$$\begin{array}{ll}\text{minimize} & f(\mathbf{y}) = (f_1(\mathbf{y}), f_2(\mathbf{y})),\\ \text{subject to} & (2)-(5).\end{array} \tag{8}$$

Our goal is to obtain Pareto solutions to this problem—hopefully, the entire Pareto front. The ε-constrained method is a well-known procedure for finding non-dominated solutions in vector optimization. Since we have only two objectives, we can implement this method quite efficiently and thus find the entire Pareto front (see, e.g., [36] and [13]). To ensure that this manuscript is self-contained, we briefly describe the method.

The ε-constrained method relies on a single objective model, keeping one of the objective functions and setting bounds on the others (by means of additional constraints). Without loss of generality, in our case we can consider the following model:

$$\begin{array}{ll}\text{minimize} & f_1(\mathbf{y}),\\ \text{subject to} & \mathbf{y} \in S,\\ & f_2(\mathbf{y}) \le \varepsilon.\end{array} \tag{9}$$

In this model, S denotes the feasibility set for the $\mathbf{y}$ vector, i.e., the set of binary vectors $\mathbf{y} \in \{0, 1\}^{|V|\times(T+1)}$, satisfying (2)–(5).

Denote by $f^1 = (f_1^1, f_2^1)$ and $f^2 = (f_1^2, f_2^2)$ two points in the criteria space such that $f_1^1 \le f_1^2$ and $f_2^1 \le f_2^2$. Using the terminology introduced by Boland et al. [13], we define $R(f^1, f^2)$ as the rectangle in the criteria space that has f^1 and f^2 as extremes of one diagonal. Let

$$\widehat{f}_{12} = \operatorname{lex}\min_{\mathbf{y}\in S}\{f_1(\mathbf{y}), f_2(\mathbf{y}) \,|\, f(\mathbf{y}) \in R(f^1, f^2)\}.$$

$\widehat{f}_{12}$ is the point in the objective space corresponding to the minimum of $f_2(\mathbf{y})$ in rectangle $R(f^1, f^2)$ chosen among the points corresponding to the minimum value of $f_1(\mathbf{y})$ also in that rectangle. In other words, $\widehat{f}_{12}$ is obtained by solving sequentially the following two optimization problems:

$$\widehat{f}_1 = \min_{\mathbf{y}\in S}\{f_1(\mathbf{y}) \,|\, f(\mathbf{y}) \in R(f^1, f^2)\}$$

and

$$\widehat{f}_2 = \min_{\mathbf{y}\in S}\{f_2(\mathbf{y}) \,|\, f(\mathbf{y}) \in R(f^1, f^2) \wedge f_1(\mathbf{y}) \le \widehat{f}_1\}.$$

Likewise, we can represent $\widehat{f}_{21}$ as the point in the rectangle $R(f^1, f^2)$ with smallest value for $f_1(\mathbf{y})$ among all points minimizing $f_2(\mathbf{y})$.

Since we are considering two objectives, we adopt the improvement of the ε-constrained method introduced by Mavrotas [36]: the so-called AUGMECON method. In particular, we consider minimizing the makespan by setting the total cost for the resources as a constraint:

$$\text{minimize} \quad \sum_{t=ES_{n+1}}^{T} t\, y_{(n+1),t} \tag{6}$$

Table 1
The payoff table.

	$f_1(\mathbf{y})$	$f_2(\mathbf{y})$
$\min f_1(\mathbf{y})$	$f_1(\mathbf{y}^*)$	$f_2(\mathbf{y}^*)$
$\min f_2(\mathbf{y})$	$f_1(\mathbf{y}^{**})$	$f_2(\mathbf{y}^{**})$

$$\text{subject to} \quad (2)-(5), \qquad \sum_{j\in V\setminus\{n+1\}} \sum_{t=\max\{0,ES_j\}}^{\min\{T-1,LS_j\}} \left(y_{jt} \sum_{\tau=t}^{t+d_j-1} \sum_{k\in K} r_{jk}\, c_{k\tau} \right) \le \varepsilon. \tag{9}$$

As explained in [36], an optimal solution to the above problem is guaranteed to be an efficient solution only if the ε-constraint is binding. This motivated the author to consider an augmented problem that in our case is the following:

$$\text{minimize} \quad \sum_{t=ES_{n+1}}^{T} t\, y_{(n+1),t} - \gamma\ s \tag{10}$$

$$\text{subject to} \quad (2)-(5), \qquad \sum_{j\in V\setminus\{n+1\}} \sum_{t=\max\{0,ES_j\}}^{\min\{T-1,LS_j\}} \left(y_{jt} \sum_{\tau=t}^{t+d_j-1} \sum_{k\in K} r_{jk}\, c_{k\tau} \right) + s = \varepsilon, \tag{11}$$

$$s \ge 0. \tag{12}$$

In the above model, s is the slack variable of the ε constraint and γ is a small factor ensuring that the slack of the ε is as high as possible. This is an elegant way of ensuring that non-supported efficient solutions are excluded when solving this single-objective model.

Two reference solutions in the Pareto front correspond to optimizing the objective functions lexicographically:

$$\mathbf{y}^{*} \in \arg\operatorname{lex}\min_{\mathbf{y}\in S}\{f_1(\mathbf{y}), f_2(\mathbf{y}) | f(\mathbf{y}) \in R((-\infty,\infty),(-\infty,\infty))\}$$

and

$$\mathbf{y}^{**} \in \arg\operatorname{lex}\min_{\mathbf{y}\in S}\{f_2(\mathbf{y}), f_1(\mathbf{y}) | f(\mathbf{y}) \in R((-\infty,\infty),(-\infty,\infty))\}.$$

These two solutions induce the so-called payoff table, presented in Table 1.

The values in the payoff table define the range of interest for the objective function $f_2(\mathbf{y})$, which is the objective that we are setting as a constraint: $[f_2(\mathbf{y}^{**}), f_2(\mathbf{y}^{*})]$. This range can be split into a number L of subintervals with breakpoints given by $\varepsilon_0 = f_2(\mathbf{y}^{**})$, $\varepsilon_1, \ldots, \varepsilon_{L-1}, \varepsilon_L = f_2(\mathbf{y}^{*})$. We consider intervals of equal length, i.e., we set

$$\varepsilon_\ell = f_2(\mathbf{y}^{**}) + \ell\, \frac{[f_2(\mathbf{y}^{*}) - f_2(\mathbf{y}^{**})]}{L}, \quad \ell = 0,\ldots,L.$$

A set of Pareto solutions can now be found starting with ε_L and solving the problem

$$\begin{array}{ll} \text{minimize} & (10) \\ \text{subject to} & (2)-(5),(11),(12), \end{array}$$

replacing ε successively with $\varepsilon_L, \varepsilon_{L-1}, \ldots, \varepsilon_0$. Since we start with the largest value for the cost—second objective function—then, every time we visit a new breakpoint and we observe a change (improvement) in the makespan—first objective function—we have just found a new Pareto solution.

Naturally, our capability for visiting the entire Pareto Front during this process depends on the specific instance being solved and the number of breakpoints assumed for the range of the second objective function. With the above information, we can now formalize the AUGMECON method applied to our problem. This is done in Algorithm 1.

Algorithm 1. Determining the Pareto Front.

1: Compute $f_1(\mathbf{y}^*)$, $f_2(\mathbf{y}^*)$, $f_1(\mathbf{y}^{**})$, $f_2(\mathbf{y}^{**})$;
2: $s \leftarrow 1$; $\mathbf{y}_1^P \leftarrow \mathbf{y}^{**}$; // First Pareto solution found
3: $\Delta \leftarrow f_2(\mathbf{y}^*) - f_2(\mathbf{y}^{**})$;
4: Compute ε_ℓ, $\forall \ell = 0,\ldots,L$
5: $\ell \leftarrow L$;
6: **while** $\ell \ge 0$ **do**
7: $\quad \varepsilon \leftarrow \varepsilon_\ell$;
8: $\quad$ Find $\mathbf{y}_\ell \in \arg\min_{\mathbf{y}} \{(10)$ s. t. $(2)-(5),(11),(12)\}$;
9: $\quad$ **if** $f_1(\mathbf{y}_\ell) < f_1(\mathbf{y}_s^P)$ **then**
10: $\quad\quad s \leftarrow s+1$; // A new Pareto Solution has been identified
11: $\quad\quad \mathbf{y}_s^P \leftarrow \mathbf{y}_\ell$;
12: $\quad$ **end if**
13: $\quad$ **repeat**
14: $\quad\quad \ell \leftarrow \ell - 1$;
15: $\quad$ **until** $\varepsilon_\ell > f_2(\mathbf{y}_s^P)$
16: **end while**

Algorithm 1 starts by setting the first Pareto solution equal to the solution inducing the upper-left corner of the payoff table. This corresponds to the largest value of resource cost. Then, step by step, we impose a smaller value for the second objective function (ε_ℓ). When checking a new breakpoint, if we do not improve the value of the first objective function, we proceed to the next breakpoint. Otherwise (line 8) we have just found a new Pareto solution. When changing the breakpoints, we also check the current value of second objective function since there is no need to check breakpoints that are larger than or equal to that value (lines 13–15).

5. A multi-objective metaheuristic for the RCPSP with time-dependent resource costs

We have designed a metaheuristic for solving the RCPSP with time-dependent resource costs in order to consider the two objectives, makespan and cost, from a real multi-objective perspective, making it possible to obtain the set of non-dominated solutions of the problem or, at least, an approximation of this set. The algorithm is based on the general purpose template of the Non-dominated Sorting Genetic Algorithm II, NSGA-II, proposed by Deb et al. [18] which is presented as an improvement of its predecessor, NSGA [47]. The general template of the NSGA-II is presented in Algorithm 2.

Algorithm 2. NSGA-II.

```
t ← 0;
P_t ← create_initial_population(N);
fast_non_dominated_sort(P_t);
crowding_distance_assignment(P_t);
while not stopping_criterion do
    Q_t ← selection_population(P_t);
    Q_t ← crossover_population(Q_t);
    Q_t ← mutation_population(Q_t);
    R_t ← P_t ∪ Q_t;
    fast_non_dominated_sort(R_t);
    crowding_distance_assignment(R_t);
    P_{t+1} ← reduce_population(R_t);
    t ← t + 1;
end while
```

This template can be used to solve different optimization problems but, previously, different structures and procedures need to be determined. Firstly, an appropriate way to encode the solutions must be designed. The encoding design is one of the most important tasks and the performance of the algorithm depends, to a large extent, on it. Then, a method to generate the initial population, sized *N*, needs to be implemented.

Before starting the main loop, it is necessary to apply two procedures that allow us to compare two different solutions and to choose the best. These procedures are, **fast_non_dominated_sort()** and **crowding_distance_assignment()**. The first sorts the individuals of the current population in different fronts. Belonging to one front or another depends on the domination rank of each solution, which is calculated in the procedure. Specifically, in the first front we can find the solutions that are not dominated by any other solution in the population; in the second front, the solutions that are dominated by one or more solutions from the first front, and so on. Therefore, solutions in front i are better than solutions in front j if $i < j$. The second procedure calculates a metric to determine the distance in the objective space among the solutions in a given front. These two procedures assign two different values to every solution: the front the solution belongs to and the crowding distance from the solution to the rest of the solutions in the corresponding front. Now, when we need to compare two solutions to determine the best one, we will choose the one belonging to the best front, and, if there is a tie, the one with a higher distance, i.e., a solution that is in a less *dense* region will be preferable.

The main loop represents the evolution process and will be carried out until the stopping criterion, which usually depends on the number of evaluations performed or the CPU time employed, being satisfied. In the current generation, the three genetic operations, that is, selection, crossover and mutation, are applied to the current population and a new population with *N* individuals is produced. Then, both populations are joined in a double-sized population R_t and, to reduce its size to the initial one, the **reduce_population()** procedure is carried out. Only the best *N* solutions in R_t will form part of the new population and to allow comparison between the solutions, it is first necessary to form the different fronts and to calculate the distances among the solutions on the fronts. These two procedures and the one to reduce the size of the population can be considered as standard and independent of the problem to which they will be applied. The details about these procedures are described in [18]. In the next sections, we detail the different features we have designed in order to implement the algorithm.

5.1. Solutions encoding

Several different encodings have been proposed to solve the RCPSP using heuristics or metaheuristics. However, the activity list representation (ALR) [25], also called permutation-based solution representation, is the most used; given that it is the most appropriate to solve this problem regardless of the type of metaheuristic used [31]. A solution is encoded as a permutation of the activities in the project where an activity always appears in the solution after its predecessors. This is an indirect representation and, to obtain the schedule, it is necessary to apply a scheduling scheme, the serial generation scheme being the most commonly applied, although the parallel generation scheme could also be applied.

This representation has later been extended in different works in order to include additional information that allows combining different ways of generating a schedule with the same activity list [26,6,7,14,20,59,15]. Moreover, the standard activity list representation has been adapted to encode the solutions of the multi-mode RCPSP [8,21,58] including information about the execution mode of the activities. Some works have also used these encodings to manage these problems (RCPSP and MM-RCPSP) considering several objectives (for example [40]).

However, we consider that when managing multiple objectives, the previous encodings are not appropriate, because they do not allow the consideration of different objectives in the construction of the schedule. In our problem, if we use the standard activity list encoding without additional information about the objectives and activities are scheduled, one by one, as soon as possible in the order given by the list, we would always prioritize the temporal objective over the economic one and it would not be appropriate in a case like this. Some authors have considered this fact and have included information that considers the different objectives in the representation. For example, Abbasi et al. [1] propose an ALR with an additional binary gene indicating the scheme used to build the schedule: serial generation scheme (that prioritizes the makespan objective) or that proposed by Ulusoy et al. [49] which could allow scheduling the activity without prioritizing the makespan objective. However, the whole schedule is built on the same criterion and this could present a drawback.

We propose an innovative encoding where solutions are represented by a double list, a list of activities and a second binary list with the criterion to be prioritized when scheduling the corresponding activity in the scheduling process.

5.1.1. Activity list with scheduling objective

Solutions are encoded with a double list, a list of activities and a binary list of the scheduling objectives of the corresponding activities. Therefore, activities are scheduled by the order given by the list. However, when an activity $j \in V$ is going to be scheduled, its corresponding scheduling objective, makespan or cost, will determine its start time, s_j. In the first case, i.e., if the scheduling objective of an activity j is the makespan, it will be scheduled from the moment where all its predecessors finish, as soon as there are enough resources to be executed, $s_j = s_j^{mak}$. Otherwise, if the scheduling objective of the activity indicates the cost, the start time of the activity will be that in the interval $[s_j^{mak}, s_j^{mak} + max\_shift_j]$ where the cost is cheaper and there are enough resources to be executed, $s_j = s_j^{cost}$. The parameter max_shift_j, i.e., the maximum delay of the activity, should be established. It could be set at the same value for all the activities, for example, $max\_shift_j = 10$ for all j, or a different value could be generated for each activity.

Example 2. In Fig. 3, we illustrate different solutions encoded with the activity list with scheduling objective representation for the project example presented in Fig. 1 and the corresponding schedules that they

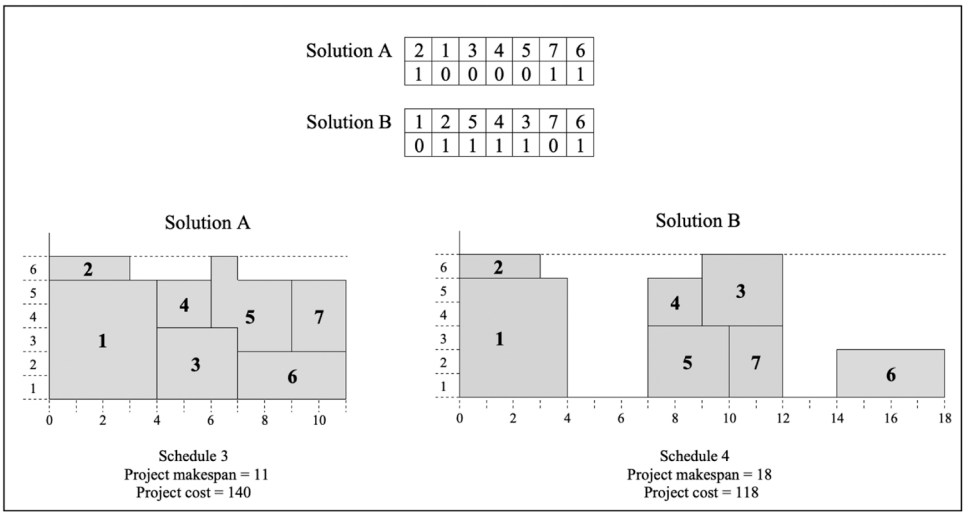


Fig. 3. Activity list with scheduling objective representation. Example.

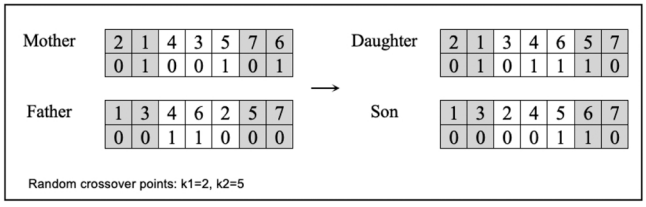


Fig. 4. Crossover example.

are transformed in. If the scheduling objective of an activity is 0 it represents the makespan and 1 indicates that the priority objective to be considered when scheduling that activity is the cost. To build the schedules, we could set, as example, $max\_shift_j = 5$ for all j. We must take into account that if an activity is scheduled when its scheduling objective refers to cost, it is scheduled in the cheapest option instead of the earliest one and, therefore, the schedule could present periods where no activity is executed because an activity has been moved to the right in the schedule in order to save money, as happens with solution B in this figure. Let us recall that, in this problem, the cost of an activity is given by the cost of the resource usage, which is dependent on the period in which they are used. For example, the cost of executing activity 3 as in schedule 3, in periods 5, 6 and 7, is 30 and the same activity, executed in periods 10, 11 and 12 as in schedule 4, costs 18.

Table 2
Evolution of the number of Pareto solutions found by AUGMECON with the number of breakpoints.

	10	30	50	70	90	110	130	150
J301_1	11	27	37	42	47	50	54	58
J3010_1	11	30	45	55	60	64	65	67
J3020_1	11	22	28	29	37	36	39	40
J3030_1	11	23	24	26	30	31	30	30
J3040_1	11	27	40	49	56	58	67	68
Average	11.0	25.8	34.8	40.2	46.0	47.8	51.0	52.6
J601_1	11	29	42	52	50	57	63	66
J6010_1	10	26	36	46	53	54	55	56
J6020_1	11	21	28	36	37	39	42	41
J6030_1	10	21	24	31	37	36	39	41
J6040_1	11	25	30	35	39	43	47	47
Average	10.6	24.4	32.0	40.0	43.2	45.8	49.2	50.2

5.2. Initial population

We have implemented a random mechanism to generate the individuals (solutions) in the initial population. To create a new solution, first, the activity list is generated. The first activity in the list is randomly chosen among the activities belonging to the eligible set, which is initially formed with the activities with no predecessors. Then, the following steps are repeated until all the activities have been chosen and occupy a position in the activity list: the eligible set is updated, including the activities with all the predecessors placed on the list; then, an activity of that set is randomly chosen to be placed in the next position on the list. Once the activity list has been generated, the scheduling objective of each activity is randomly chosen (cost or makespan) with a probability of 50% each.

5.3. Selection

The selection mechanism is applied over the current population to form a new one with the same population size. That population becomes the current population replacing the original one. Next, the crossover mechanism will be applied to it. We have implemented the selection mechanism proposed by Deb et al. [18] in the NSGA-II general template. The mechanism is based on the standard binary tournament selection. To build the new population, the following procedure is repeated until the new population is fulfilled: two individuals of the current population are randomly chosen and they compete for a place in the new population; the winner of this tournament is the one belonging to the best front or, in case of ties, the one with a best crowding distance. The details of this mechanism can be consulted in [18].

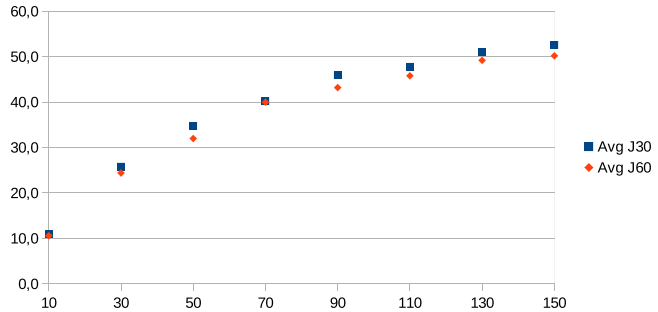


Fig. 5. Average number of Pareto solutions.

5.4. Crossover

The current population resulting from the selection procedure undergoes the crossover mechanism to build a new population where the mutation procedure will be applied. Pairs of individuals are randomly chosen and, if they undergo the crossover operation, two offspring are produced and are included in the new population. Otherwise, the selected individuals are included in the new population. The parameter to decide if a pair of individuals are combined to generate offspring or not is the crossover probability, *Pcross*, which does not depend on the pair of solutions to be crossed.

The crossover operator is applied over a pair of solutions and it should combine the information of both solutions, the parents, in order to create the offspring. Since in our case the solutions are represented by a list of activities and a list of scheduling objectives, the procedure must be designed to manage and combine the information of both lists. The crossover operator we have designed has two phases: first, the activity lists of the parents are combined and, in a second phase, the offspring inherit the information contained in the scheduling objective lists.

The first phase consists on applying the two-point crossover proposed by Hartmann [25] in his genetic algorithm to solve the RCPSP in order to cross the activity lists of the parents. This was designed as a two-point crossover operator applied to permutations but taking into account that an activity can only appear in the activity list after all its predecessors. Therefore, this crossover always generates feasible lists. In a second phase, the activities in the offspring inherit the scheduling objectives present in the parent from which the activity was inherited.

Example 3. We illustrate the above procedure in Fig. 4, where two solutions for the project presented in Fig. 1, the mother and the father, have undergone the crossover operation. First, two random crossover points, for example, $k_1 = 2$ and $k_2 = 5$ are generated, dividing the list into three parts. Then, one of the offspring, the daughter, inherits the first two positions in the activity list from the mother. The following three are inherited from the father: we look for the first three activities in the father not present in the daughter, maintaining the relative order between them in the father. The last two activities are those which are not present in the daughter and maintain their relative order in the mother. The activity list in the son is obtained in the same way, but interchanging the role of the parents. In the second phase, activities inherit the scheduling mode they had in the parent from which they have been copied. In this way, the offspring always represent feasible schedules because, applying this procedure, an activity can not appear in the list before any of its predecessors.

5.5. Mutation

The mutation mechanism is applied to every individual in the current population and, if the solution mutates, it replaces the original one. The mutation mechanism allows more variability in the population to be introduced, it can include new characteristics in one or more individuals of the population or characteristics that were present in the past but have been lost during the evolution process. We have designed a mutation operator that is applied to every individual and, as the crossover operator, consists of two phases: the first is applied to the activity list and the second to the scheduling objective list.

The mechanism applied in the first phase is the procedure employed by Alcaraz and Maroto [6] in their GA to solve the RCPSP, which was first proposed by Boctor [12] to generate neighbors in his simulated annealing algorithm to solve the problem. Following the authors, for each activity in the sequence, a new position is randomly chosen, between the last of its predecessors and the first of its successors which ensures the generation of only precedence feasible solutions. The activity is inserted in the new position with a probability of *Pmut_act*. After the insertion, all the activities in the list maintain their scheduling objective. This procedure allows the order in which the activities will be chosen in the scheduling scheme to be changed. In the second phase, the scheduling objective of each activity changes with a probability of *Pmut_obj*. The scheduling objective changes from 0 to 1 or vice versa. This second phase allows the objective considered to schedule an activity in the scheduling process carried out to build the corresponding schedule to be changed.

6. Empirical analysis

In this section, we report on the computational tests performed to assess the methodological contribution of this paper. We start by describing the test bed instances used. Afterwards we provide the details regarding the experimental setting adopted both for AUGMECON and for the metaheuristic. In particular, we discuss the different metrics that were considered for assessing the approximate Pareto front provided by the metaheuristic. We then present results for AUGMECON for the instances for which we are sure to have found only exact Pareto solutions. Finally, we analyze the other instances—those for which it was not possible to find the exact front.

6.1. Test data

In order to obtain test data for the methodologies proposed above, we consider the instances available in the PSPLIB library (http://www.om-db.wi.tum.de/psplib/) as a basis. In particular, we considered the single mode data sets J30, J60, J90 and J120, where the number indicates the number of activities that each instance (project) in the set has. The first three sets have 480 instances each and J120 has a total of 600 instances.

The instances were generated by an automatic generator, combining three different factors: Network complexity (NC), resource factor (RF), and resource strength (RS) (see [32]). The first corresponds to the average number of direct successors of the activities. The second measures the average proportion of resources required by each activity and is a value between 0 and 1 where a value close to 1 indicates that the activities are very resource-demanding. Finally, the resource strength regards the mean tightness of the resource constraints and like the RF, this is also a value in the interval [0,1] with a value close to 1 indicating that the available resources are enough to allow all the activities to start at their earliest starting time; on the other hand, a value of RS close to zero stems from an instance with scarce resources—each activity calls for the usage of resources in the limit (or close to it).

In the first three datasets, we have three different levels for NC (1.5, 1.8 and 2.1), four levels for RF (0.25, 0.5, 0.75 and 1) and four for the parameter RS (0.2, 0.5, 0.7 and 1). The combination of these levels gives a total of $3 \cdot 4 \cdot 4 = 48$ combinations and for each combination, ten different instances were generated, what means that each one of these datasets has 480 instances. The instances in these sets in PSPLIB are named as JXY_Z, where $X \in \{30, 60, 90\}$ indicates the number of activities each project has, $Y \in \{1, \ldots, 48\}$ represents the number of combination of the parameters and $Z \in \{1, \ldots, 10\}$ the number of

repetitions. In the case of J120, we can find the same levels for NC and RF as in the first three datasets, but there are five RS levels (0.1, 0.2, 0.3, 0.4 and 0.5) instead of four. Therefore, in J120 we have $3 \cdot 4 \cdot 5 = 60$ combinations for the factors and for each one we can find 10 repetitions which give 600 instances. These instances are named as J120Y_Z, for $Y \in \{1, \ldots, 60\}$ and $Z \in \{1, \ldots, 10\}$.

As the number of instances is very high, first we have selected a total of 48 instances with 30 activities and the same number of instances with 60 activities. In order to have instances with all the combinations of factors, we have selected the first repetition of each group, i.e., JXY_1. As for the larger instances (90 and 120 activities), very early in our work we realized that the use of the exact algorithm would be tantalizing and thus, instead of considering all the instances, we report results for only a few in each set, namely: J901_1, J9010_1, J9020_1, J9030_1, J9040_1 for 90 activities, and J1201_1, J12010_1, J12020_1, J12030_1, J12040_1, J12050_1, J12060_1 for 120 activities.

For each instance, we set T as the sum of the processing times of all the activities. Moreover, none of the above instances include time-dependent costs for the resources and thus, that component of the data was generated for this work. The methodology is presented next.

We can consider a general pattern for the evolution of the cost of a resource throughout time. If c_t is the cost for a resource at time t (valid from time t to time $t+1$) we set

$$c_t = \alpha + \beta t + \gamma_t + \omega_t, \quad t = 0, \ldots, T-1,$$

with.

- α, representing a constant defining a base level of the cost series;
- β, representing a constant slope determining a cost trend;
- γ_t, representing the seasonal term for the time period starting at time t;
- ω_t representing a random variable such that $\mathbb{E}[\omega_t] = 0$ and $\mathbb{V}[\omega_t] = \sigma_\omega^2$.

If L is the length of a season (number of time periods), then there are at most L seasonality different terms, that will repeat throughout time. We can also assume that

$$\gamma_0 + \gamma_1 + \cdots + \gamma_{L-1} = \Gamma \times L.$$

More generally, for $t \geq L$ (and integer) we assume that

$$\gamma_{t-L} + \gamma_{t-L+1} + \cdots + \gamma_{t-1} = \Gamma \times L.$$

These assumptions are justified by the fact that the seasonal terms represent a deviation above and below some average, Γ. Thus, the average of any L consecutive seasonal terms should always be equal to Γ.

From here we can consider four patterns for the evolution of a resource cost:

Pattern 1: trend with a positive slope; no seasonality.
Pattern 2: trend with a negative slope; no seasonality.
Pattern 3: trend with a positive slope; with seasonality.
Pattern 4: trend with a negative slope; with seasonality.

Given that the instances available in PSPLIB for the resource-constrained project scheduling problem contain 4 resources each, we assigned one pattern to each resource in that order: resource 1 → pattern 1,., resource 4 → pattern 4.

Table 3
Instances J30 and J60: features and resolution using AUGMECON.

			J30		J60	
NC	RF	RS	Instance	Optimal front?	Instance	Optimal front?
1.5	0.25	0.2	J301_1	✓	J601_1	✓
		0.5	J302_1	✓	J602_1	✓
		0.7	J303_1	✓	J603_1	✓
		1	J304_1	✓	J604_1	✓
	0.5	0.2	J305_1	✓	J605_1	—
		0.5	J306_1	✓	J606_1	—
		0.7	J307_1	✓	J607_1	—
		1	J308_1	✓	J608_1	✓
	0.75	0.2	J309_1	—	J609_1	—
		0.5	J3010_1	✓	J6010_1	—
		0.7	J3011_1	—	J6011_1	—
		1	J3012_1	✓	J6012_1	✓
	1	0.2	J3013_1	—	J6013_1	—
		0.5	J3014_1	—	J6014_1	—
		0.7	J3015_1	✓	J6015_1	✓
		1	J3016_1	✓	J6016_1	✓
1.8	0.25	0.2	J3017_1	✓	J6017_1	—
		0.5	J3018_1	✓	J6018_1	—
		0.7	J3019_1	✓	J6019_1	—
		1	J3020_1	✓	J6020_1	✓
	0.5	0.2	J3021_1	✓	J6021_1	—
		0.5	J3022_1	✓	J6022_1	—
		0.7	J3023_1	✓	J6023_1	✓
		1	J3024_1	✓	J6024_1	✓
	0.75	0.2	J3025_1	—	J6025_1	—
		0.5	J3026_1	✓	J6026_1	—
		0.7	J3027_1	✓	J6027_1	—
		1	J3028_1	✓	J6028_1	—
	1	0.2	J3029_1	—	J6029_1	—
		0.5	J3030_1	—	J6030_1	—
		0.7	J3031_1	✓	J6031_1	—
		1	J3032_1	✓	J6032_1	—
2.1	0.25	0.2	J3033_1	✓	J6033_1	—
		0.5	J3034_1	✓	J6034_1	—
		0.7	J3035_1	✓	J6035_1	—
		1	J3036_1	✓	J6036_1	✓
	0.5	0.2	J3037_1	—	J6037_1	—
		0.5	J3038_1	✓	J6038_1	—
		0.7	J3039_1	✓	J6039_1	✓
		1	J3040_1	✓	J6040_1	—
	0.75	0.2	J3041_1	—	J6041_1	—
		0.5	J3042_1	✓	J6042_1	—
		0.7	J3043_1	—	J6043_1	—
		1	J3044_1	✓	J6044_1	✓
	1	0.2	J3045_1	—	J6045_1	—
		0.5	J3046_1	—	J6046_1	—
		0.7	J3047_1	✓	J6047_1	—
		1	J3048_1	✓	J6048_1	—

For a resource (and for the corresponding cost pattern) we generate the costs as follows:

- A cost level α is generated randomly according to a uniform distribution $U[100,200]$.
- The slope is generated in such a way that if no perturbation exists, then the level of the series in the last time would be equal to $\alpha/2$. A minimum (maximum) slope of 0.1 (−0.1) is imposed. Thus, for getting a positive (negative) slope, β is randomly generated according to a uniform distribution $U[0.1, \frac{\alpha}{2T}]$ ($U[-\frac{\alpha}{2T}, -0.1]$) if $\frac{\alpha}{2T} > 0.1$; β is set equal to 0.1 (−0.1) otherwise.
- For the patterns with seasonality, we set $L = 12$ (12 weeks—three months; 12 months; ...). We generate Γ randomly according to a

Table 4
Number of optimal Pareto fronts found.

(a) Instances with 30 activities.

	RS= 0.2			RS= 0.5			RS= 0.7			RS= 1.0			**Total**
	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	**(RF)**
RF= 0.25	1	1	1	1	1	1	1	1	1	1	1	1	**12**
RF= 0.50	1	1	0	1	1	1	1	1	1	1	1	1	**11**
RF= 0.75	0	0	0	1	1	1	0	1	0	1	1	1	**7**
RF= 1.0	0	0	0	0	0	0	1	1	1	1	1	1	**6**
Total (NC)	**2**	**2**	**1**	**3**	**3**	**3**	**3**	**4**	**3**	**4**	**4**	**4**	
Total (RS)	**5**			**9**			**10**			**12**			

(b) Instances with 60 activities.

	RS= 0.2			RS= 0.5			RS= 0.7			RS= 1.0			**Total**
	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	NC= 1.5	NC= 1.8	NC= 2.1	**(RF)**
RF= 0.25	1	0	0	1	0	1	1	1	1	1	1	1	**9**
RF= 0.50	0	0	0	0	0	0	0	1	1	1	1	1	**5**
RF= 0.75	0	0	0	0	0	0	0	0	0	1	1	1	**3**
RF= 1.0	0	0	0	0	0	0	1	0	0	1	0	0	**2**
Total (NC)	**1**	**0**	**0**	**1**	**0**	**1**	**2**	**2**	**2**	**4**	**3**	**3**	
Total (RS)	**1**			**2**			**6**			**10**			

uniform distribution $U[20,30]$. The different seasonality terms that will repeat throughout time are:

$$0,\ \Gamma,\ 2\Gamma,\ 3\Gamma,\ 2\Gamma,\ \Gamma,\ 0,\ -\Gamma,\ -2\Gamma,\ -3\Gamma,\ -2\Gamma,\ -\Gamma.$$

- The noise ω_t is generated according to a Normal distribution $N(0, 5)$.

6.2. Experimental setting

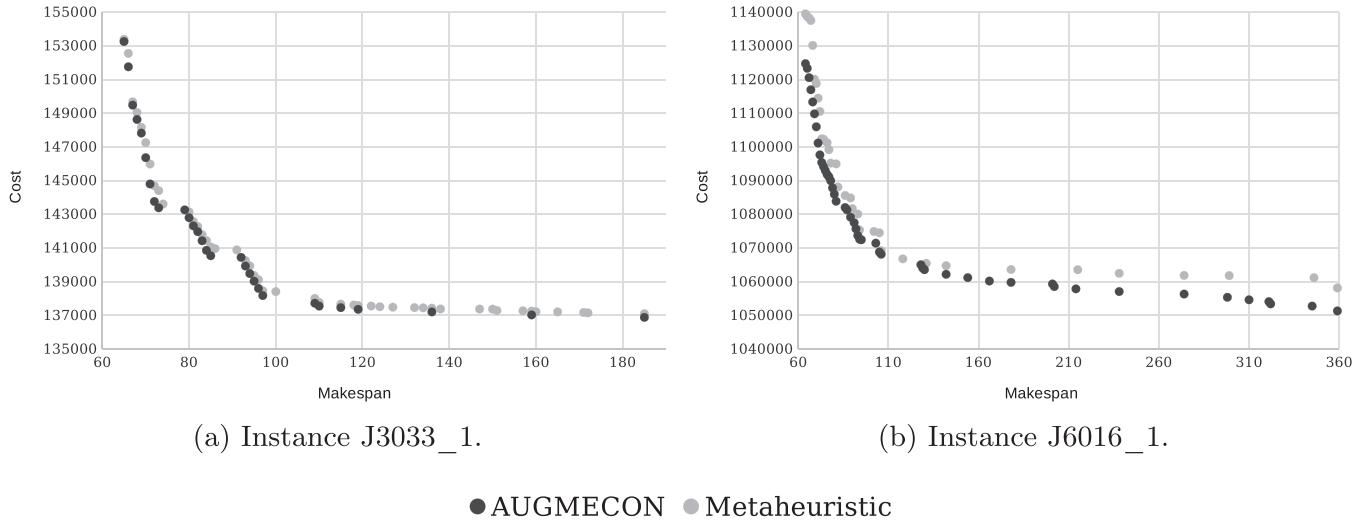
We now detail several aspects defining the experimental setting considered in this work.

Table 5
Metrics and CPU time for the J30 instances such that AUGMECON could successfully solve all the MILP problems.

	AUGMECON				Metaheuristic							
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	HVR	IGD+	Spread	Time (hrs)
J301_1	49	100%	0.087	1.48	63	1.351	0.074	0.094	95.53%	0.024	0.545	0.57
J302_1	59	98.15%	0.067	1.28	54	1.362	0.09	0.051	94.08%	0.025	0.502	0.53
J303_1	55	100%	0.101	0.58	40	1.365	0.132	0.075	91.79%	0.033	0.591	0.85
J304_1	42	100%	0.097	0.36	34	1.378	0.097	0.096	88.61%	0.049	0.63	0.51
J305_1	57	100%	0.088	13.5	56	1.385	0.088	0.045	94.6%	0.028	0.464	0.61
J306_1	40	100%	0.155	7.8	35	1.252	0.155	0.061	95.07%	0.027	0.644	0.59
J307_1	34	100%	0.121	0.37	30	1.374	0.111	0.06	95.18%	0.027	0.638	0.65
J308_1	44	100%	0.117	0.5	34	1.388	0.124	0.077	92.13%	0.037	0.503	0.55
J3010_1	60	100%	0.082	6.79	56	1.373	0.107	0.095	91.53%	0.039	0.521	0.63
J3012_1	37	100%	0.099	0.99	32	1.359	0.198	0.102	90.32%	0.037	0.677	0.63
J3015_1	41	100%	0.125	2.69	35	1.528	0.143	0.063	91.44%	0.042	0.609	0.59
J3016_1	43	100%	0.108	1.71	32	1.395	0.127	0.114	89.02%	0.046	0.569	0.51
J3017_1	59	98.39%	0.078	12.16	62	1.384	0.061	0.052	95.08%	0.024	0.475	0.6
J3018_1	47	100%	0.093	0.73	39	1.353	0.111	0.093	93.27%	0.031	0.626	0.53
J3019_1	51	95.65%	0.078	0.49	46	1.313	0.098	0.096	93.07%	0.032	0.497	0.55
J3020_1	36	100%	0.113	0.64	29	1.339	0.114	0.113	92.15%	0.048	0.591	0.71
J3021_1	42	100%	0.129	19.62	37	1.392	0.118	0.071	94.6%	0.028	0.553	0.59
J3022_1	35	100%	0.115	2.25	35	1.382	0.115	0.055	96.44%	0.018	0.61	0.46
J3023_1	41	100%	0.092	0.93	33	1.369	0.092	0.078	93.14%	0.03	0.565	0.68
J3024_1	21	100%	0.348	0.19	19	1.229	0.214	0.051	97.46%	0.023	0.783	0.68
J3026_1	26	100%	0.234	16.75	22	1.391	0.213	0.129	84.79%	0.066	0.588	0.54
J3027_1	29	100%	0.48	6.81	29	1.257	0.153	0.102	92.91%	0.042	0.531	0.59
J3028_1	20	100%	0.529	0.63	18	1.193	0.173	0.057	96.49%	0.023	0.582	0.56
J3031_1	22	100%	0.375	8.83	23	1.442	0.375	0.079	94.64%	0.033	0.611	0.44
J3032_1	17	100%	0.489	0.37	20	1.507	0.267	0.069	94.88%	0.031	0.539	0.91
J3033_1	29	100%	0.217	2.79	46	1.41	0.176	0.026	98.31%	0.011	0.547	0.74
J3034_1	20	100%	0.289	0.56	16	1.396	0.289	0.065	91.92%	0.042	0.445	0.75
J3035_1	12	100%	0.412	0.11	11	1.347	0.471	0.103	88.51%	0.057	0.585	0.55
J3036_1	27	84.62%	0.304	0.21	26	1.251	0.417	0.031	99.32%	0.005	0.852	0.73
J3038_1	49	100%	0.103	2.38	42	1.358	0.112	0.1	93.66%	0.034	0.589	0.44
J3039_1	53	100%	0.089	0.99	45	1.368	0.107	0.09	90.58%	0.041	0.647	0.6
J3040_1	62	100%	0.089	1.12	45	1.334	0.109	0.123	87.9%	0.061	0.6	0.68
J3042_1	41	94.44%	0.215	17.03	36	1.406	0.206	0.046	96.36%	0.022	0.624	0.62
J3044_1	29	100%	0.237	0.8	24	1.409	0.495	0.069	95.19%	0.026	0.778	0.54
J3047_1	39	100%	0.126	6.15	36	1.405	0.147	0.061	93.42%	0.03	0.535	0.48
J3048_1	49	100%	0.127	0.71	39	1.352	0.139	0.128	89.76%	0.045	0.596	0.46
Average	39.4	–	–	3.9	35.5	–	–	–	–	–	–	0.6

Table 6
Metrics and CPU time for the J60 instances such that AUGMECON could successfully solve all the MILP problems.

	AUGMECON				Metaheuristic							
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	HVR	IGD+	Spread	Time (hrs)
J601_1	56	100%	0.078	23.22	51	1.425	0.18	0.204	83.29%	0.08	0.708	1.83
J602_1	84	100%	0.054	32.71	68	1.253	0.127	0.295	76.52%	0.111	0.685	1.56
J603_1	74	100%	0.046	12.85	80	1.364	0.11	0.167	89.81%	0.051	0.677	1.65
J604_1	29	100%	0.224	6.51	17	0.971	0.253	0.218	81.63%	0.069	0.747	2.07
J608_1	80	100%	0.054	14.12	62	1.306	0.095	0.194	81.05%	0.087	0.543	2.11
J6012_1	77	100%	0.04	18.78	55	1.321	0.092	0.193	79.18%	0.09	0.623	1.38
J6015_1	54	100%	0.096	18.87	28	1.437	0.201	0.197	82.29%	0.084	0.662	2.04
J6016_1	47	97.06%	0.122	44.22	34	1.493	0.159	0.106	91.87%	0.036	0.667	2.87
J6020_1	36	100%	0.172	20.69	26	1.587	0.396	0.146	87.6%	0.061	0.754	2.13
J6023_1	76	100%	0.049	26.21	59	1.345	0.135	0.201	84.88%	0.069	0.593	2.25
J6024_1	39	100%	0.119	2.37	26	1.057	0.099	0.131	90.81%	0.041	0.733	1.41
J6036_1	36	100%	0.181	1.83	31	1.352	0.34	0.158	88.79%	0.062	0.857	1.4
J6039_1	30	100%	0.545	42.73	20	1.075	0.145	0.108	89.53%	0.05	0.732	1.57
J6044_1	80	100%	0.052	15.36	39	1.358	0.126	0.219	79.57%	0.085	0.587	1.56
Average	57.0	–	–	20.0	42.6	–	–	–	–	–	–	1.8



(a) Instance J3033_1. (b) Instance J6016_1.

Fig. 6. Exact and heuristic Pareto solutions for 2 selected instances.

Table 7
Metrics and CPU time for the J30 instances such that AUGMECON could not successfully solve all the MILP problems.

	AUGMECON				Metaheuristic				
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	Time (hrs)
J309_1	26	87.5%	0.137	80.21	32	1.346	0.151	0.156	0.46
J3011_1	44	97.06%	0.117	36.67	34	1.348	0.233	0.059	0.53
J3013_1	22	53.57%	0.178	44.1	28	1.427	0.274	0.094	0.46
J3014_1	48	100%	0.088	56.93	36	1.298	0.099	0.089	0.44
J3025_1	8	26.47%	0.355	84.09	34	2.677	0.387	0.051	0.62
J3029_1	19	72.41%	0.26	68.2	29	1.861	0.211	0.11	0.62
J3030_1	25	90.63%	0.169	54.68	32	1.468	0.145	0.092	0.48
J3037_1	38	100%	0.11	45.53	43	1.424	0.11	0.046	0.48
J3041_1	12	52.94%	0.357	60.66	34	1.684	0.179	0.104	0.52
J3043_1	30	100%	0.207	30.87	30	1.278	0.14	0.05	0.64
J3045_1	12	26.09%	0.208	58.07	23	1.628	0.218	0.098	0.4
J3046_1	44	97.44%	0.109	79.33	39	1.322	0.13	0.07	0.46
Average	27.3	–	–	58.3	32.8	–	–	–	0.5

Table 8
Metrics and CPU time for the J60 instances such that AUGMECON could not successfully solve all the MILP problems.

	AUGMECON				Metaheuristic				
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	Time (hrs)
J605_1	6	62.16%	0.479	46.82	37	2.57	0.358	0.176	1.29
J606_1	20	100%	0.286	79.28	28	2.928	0.714	0.142	1.38
J607_1	11	71.43%	0.289	95.92	21	5.203	1.167	0.167	2.31
J609_1	23	64.86%	0.39	74.76	74	1.776	0.115	0.132	1.13
J6010_1	44	100%	0.079	132.26	58	1.416	0.106	0.223	1.5
J6011_1	89	100%	0.034	80.07	74	1.331	0.086	0.194	1.59
J6013_1	1	2.04%	—	200	49	—	—	—	2
J6014_1	19	80.56%	0.189	74.09	36	3.865	0.83	0.105	2.58
J6017_1	42	100%	0.106	68.15	43	1.401	0.335	0.136	1.48
J6018_1	16	52.78%	0.278	78.43	36	6.419	0.667	0.14	2.22
J6019_1	14	75%	0.883	42.87	24	1.474	0.307	0.085	1.85
J6021_1	15	48.44%	0.612	60.19	64	1.735	0.152	0.13	2
J6022_1	44	100%	0.101	75.36	46	1.443	0.101	0.093	2.1
J6025_1	1	16.98%	—	200	53	—	—	—	1.82
J6026_1	25	100%	0.177	86.21	34	2.081	0.194	0.164	2.45
J6027_1	10	77.27%	0.381	105.14	22	2.883	0.429	0.286	2.47
J6028_1	29	100%	0.56	11.86	20	1.356	0.192	0.087	2.72
J6029_1	4	38.98%	0.54	50.23	59	3.348	0.276	0.124	1.97
J6030_1	31	100%	0.208	61.11	40	1.319	0.104	0.104	1.48
J6031_1	40	100%	0.161	112.27	34	1.488	0.426	0.108	1.5
J6032_1	33	91.67%	0.347	71.49	24	1.739	0.32	0.093	2.26
J6033_1	21	100%	0.545	58.11	17	1.204	0.176	0.114	1.28
J6034_1	10	47.22%	0.313	85.62	36	5.534	0.813	0.125	1.62
J6035_1	27	86.67%	0.24	39.14	30	0.984	0.165	0.095	1.56
J6037_1	1	19.35%	—	200	31	—	—	—	1.58
J6038_1	16	100%	0.747	117.76	30	1.269	0.18	0.123	1.62
J6040_1	40	100%	0.168	28.39	25	0.993	0.147	0.173	1.9
J6041_1	2	19.05%	1	57.24	63	2.974	0.236	0.186	1.28
J6042_1	50	96.25%	0.123	57.06	80	1.397	0.083	0.149	1.56
J6043_1	36	100%	0.331	70.69	28	1.262	0.087	0.161	1.37
J6045_1	1	3.45%	—	200	58	—	—	—	1.04
J6046_1	11	94.74%	0.252	52.34	38	1.742	0.18	0.105	1.61
J6047_1	14	78.26%	0.2	136.09	23	3.242	0.6	0.25	1.84
J6048_1	17	78.95%	0.257	85.74	19	1.868	0.314	0.143	1.69
Average	22.4	–	–	88.1	39.8	–	–	–	1.8

6.2.1. AUGMECON

The AUGMECON method calls for solving a sequence of MILP problems for each instance. The algorithm was coded in C++ and integrated with IBM CPLEX 20.1 through Concert Technology.

A first important decision concerning the use of AUGMECON concerns the number of breakpoints to consider to split the cost range. We recall that the cost range is obtained from the costs in the payoff table. It is important to note that setting a given number n of breakpoints leads AUGMECON to obtain a maximum of $n + 1$ points on the front, given that the cost range is divided into $n + 1$ subintervals.

Since we had no hint about the number of breakpoints that should be considered, we conducted preliminary experiments using a small subset of instances namely: J301_1, J3010_1, J3020_1, J3030_1, J3040_1, J601_1, J6010_1, J6020_1, J6030_1 and J6040_1. We applied AUGMECON several times for each instance using a different number of breakpoints: 10, 30, 50, 70, 90, 110, 130 and 150. In these experiments we set a time limit of 2 h for each MILP solver, i.e., two hours for each non-dominated solution, and we left all the other parameters as default. The results obtained are summarized in Table 2 and depicted in Fig. 5.

In Fig. 5, we observe a tendency for the average number of breakpoints to stabilize around values 90 and 100. In fact, although a growing trend can still be observed after 100 breakpoints are considered, the differences from a number of breakpoints to the following seem to decrease. Given that we needed to seek a comfortable trade-off between the number of breakpoints to consider and the computing effort when computing the Pareto front, we adopted the round figure '100' as the 'stabilizing' point. For this reason, for the J30 and J60 instances, we decided to use 100 breakpoints. For the larger instances, given the predictable additional computational effort to solve the MILP models, we decided to reduce the above number to 50. Furthermore, we keep considering a time limit of 2 h for solving the MILP models associated with the smaller instances (30 and 60 activities). For the larger instances we consider 4 h. Note that this time limit is set for each MILP solved and thus for analyzing each breakpoint.

6.2.2. Metaheuristic

In order to evaluate the performance of the metaheuristic designed, we have implemented it and we have solved the instances in PSPLIB described above with this technique. We have implemented the metaheuristic proposed using the jMetal framework [19,41], which is a widely used open-source framework for multi-objective optimization with metaheuristics (see, e.g., [34,33,56,57]).

We carried out some preliminary experiments to set the best configuration of the algorithm in the different scenarios. Firstly, we combined different values for the parameters *Pcross*, *Pmut_act*, *Pmut_obj* and Population size. Although there was not a combination of values that performed the best in all the experiments, we decided to set a fixed configuration for all the runs in order to avoid a custom configuration

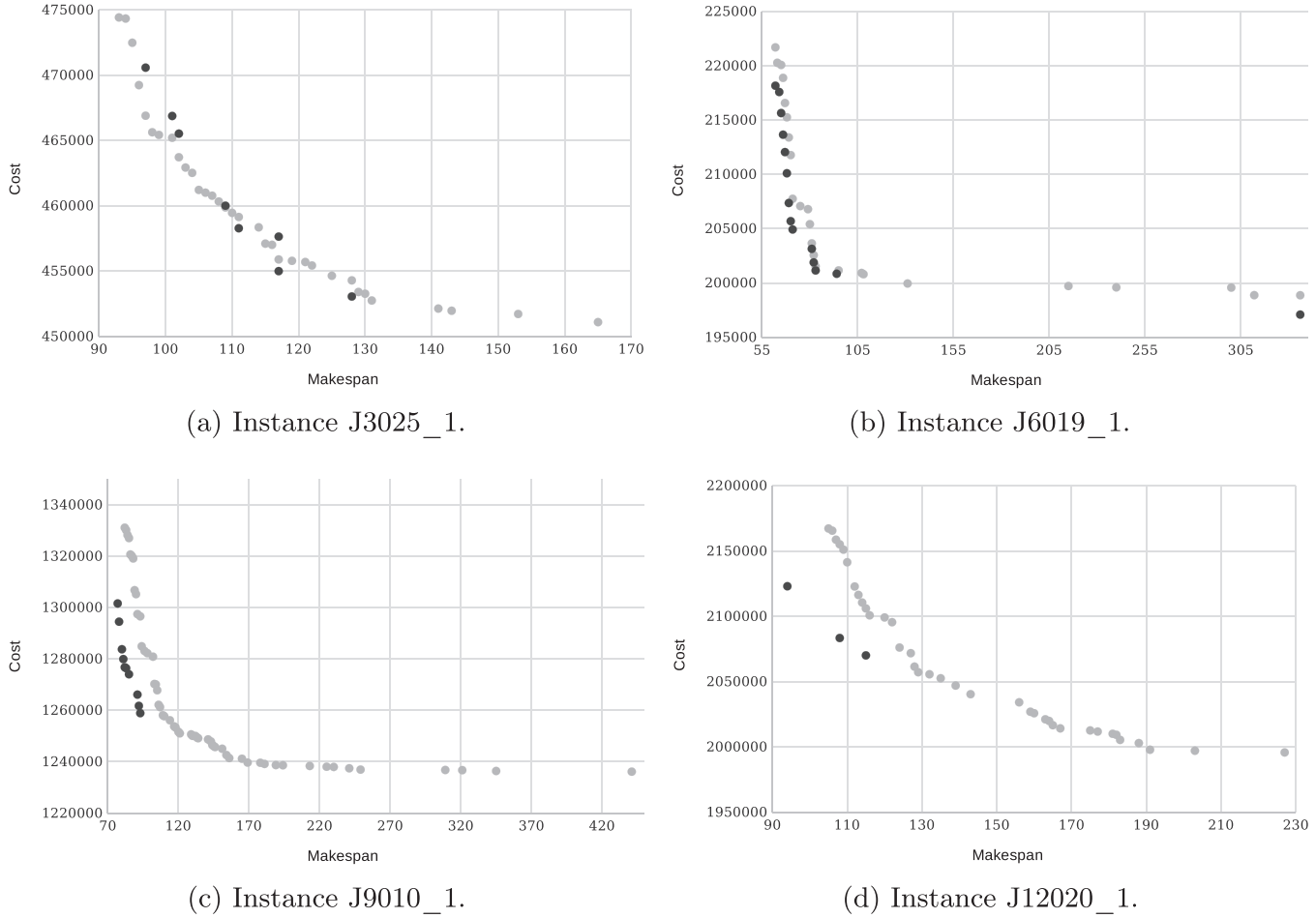
Table 9
Metrics and CPU time for the J90 instances such that AUGMECON could not successfully solve all the MILP problems.

	AUGMECON				Metaheuristic				
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	Time (hrs)
J901_1	13	82.76%	0.24	104.3	87	1.831	0.253	0.079	3.94
J9010_1	10	36.36%	0.375	130.77	55	22.547	6	0.599	3.16
J9020_1	13	60%	0.222	88.42	25	4.842	1.333	0.213	2.93
J9030_1	2	71.43%	1	88.07	21	8.266	1.333	0.778	5.73
J9040_1	8	45.83%	0.385	180.91	24	5.858	0.714	0.357	3.29
Average	9.2	–	–	118.5	42.4	–	–	–	3.8

for each instance. After these parameters were set, we ran the different instances with different numbers of iterations and 20 million seemed a good trade-off between the computational time employed and the quality of the results. Therefore, the following combination of these parameters was set in all the runs:

- $Pcross = 0.9$
- $Pmut\_act = Pmut\_obj = 1/n$; being n the number of activities in the project
- Population size: 100
- Total number of evaluations: 20 million

The parameter max_shift_j, for all $j \in V$, must be set before the evaluation of each solution in order to allow the transformation of an individual in the population into a schedule. After some preliminary experiments, we set an upper bound of $T/2$ for this maximum delay of each activity. Next, we performed some test experiments setting the same value of the parameter max_shift_j for all activities in the project, as well as obtaining a different parameter value for each activity. We also analyzed how the behavior of the metaheuristic was influenced by the fact of always randomly generating the parameter max_shift_j in the interval [1, $T/2$], or allowing this parameter to be progressively larger, that is, allowing that, as the metaheuristic search process evolves, the activities can be more delayed. After some test experiments, we ended up defining four different strategies for setting the maximum shift for the activities in the project when a schedule is to be built (evaluation performed):



(a) Instance J3025_1.

(b) Instance J6019_1.

(c) Instance J9010_1.

(d) Instance J12020_1.

Fig. 7. Approximate Pareto solutions for 4 selected instances.

- Strategy 1. A unique random integer number, *max_shift*, is generated in the interval [1, $T/2$] and $max\_shift_j = max\_shift$ for all $j \in V$.
- Strategy 2. For each $j \in V$, a random integer number, max_shift_j, is generated in the interval [1, $T/2$].
- Strategy 3. A unique random integer number, *max_shift*, is generated in the interval [A, B] and $max\_shift_j = max\_shift$ for all $j \in V$. Now, the interval where to choose the parameter depends on the number of evaluations performed so far. In the first 10% of the evaluations, [A, B] = [1, $T/8$], in the next 20% [A, B] = [$T/8 + 1$, $T/4$], in the following 30% of evaluations [A, B] = [$T/4 + 1$, $3T/8$] and in the last 40% [A, B] = [$3T/8 + 1$, $T/2$].
- Strategy 4: For each $j \in V$, a random integer number, max_shift_j, is generated in the interval [A, B]. The interval [A, B] is formed as in strategy 3 and it depends on the number of schedules built so far.

In strategies 3 and 4, all the intervals have the same length and the percentage of evaluations considered to determine the corresponding interval has been set after some test experiments. Moreover, preliminary experiments showed that none of these criteria performed the best in all the instances. On the other hand, it was possible to find that each one of the criteria performed the best in one or more instances. Therefore, we decided to combine the four criteria in the resolution of every instance. Given that the maximum number of evaluations is set to 20 million per instance, we have performed 4 independent runs employing one of the four criteria in each, setting a total of 5 million evaluations per run. The non-dominated solutions of each run are included in a set and the result of the metaheuristic is formed with the non-dominated solutions of this set, which forms a front.

6.2.3. Metrics for evaluating the approximate Pareto front

A good review of metrics to measure the quality of Pareto front approximations in multi-objective optimization can be found in [10]. The authors classify the metrics according to their properties: cardinality, convergence, distribution and spread. *Cardinality indicators* quantify the number of non-dominated points generated by an algorithm. *Convergence indicators* quantify how close a set of non-dominated points is from the Pareto front in the objective space. *Distribution and spread indicators* quantify the distribution of a Pareto front approximation. Coverage measures how well every region of the objective space is represented, while spread focuses on the aspect that points should be far away from each other. There are also some *convergence and distribution indicators* which capture both the properties of convergence and distribution. Following this classification by Audet et al. [10], we detail below the metrics used in this work. The selection of the metrics has been made in such a way that all the categories are covered and, moreover, the information given by them all give us a detailed description of the characteristics that the front has. Moreover, the metrics have been selected because they are easy to interpret and have been widely used in the literature.

For instances in which the optimal Pareto front cannot be obtained, we decided to calculate the following metrics to compare two fronts: .

- Cardinality:
  - Overall non-dominated Vector Generation (OVNG), proposed by van Veldhuizen and Lamont [50]: returns the number of non-dominated points on the front.
  - C-metric, proposed by Zitler and Thiele [61]: gives for two fronts, F_1 and F_2, the fraction of solutions in F_1 that are dominated by one or more solutions in F_2, $C(F_1, F_2)$.

- Distribution and spread:
  - Γ-metric, proposed by Custòdio et al. [17]: when considering a bi-objective problem, reduces to consider the maximum distance between two consecutive points in the Pareto front approximation, therefore, a lower value of Γ is desirable.
  - M^*_3-metric, proposed by Zitler et al. [60]: in the case of two objectives, this equals the distance of the two outer solutions and, consequently, a higher distance is desired.

- Convergence:
  - ϵ-indicator, proposed by Zitzler et al. [63]: gives the minimum additive factor by which the approximation set has to be translated in the objective space in order to (weakly) dominate the reference set. A lower value is desirable.

In addition to the above metrics, for the instances where the optimal Pareto front is known, we also calculate the following metrics: .

- Convergence and distribution:
  - Hypervolume ratio, HVR, proposed by Zitzler [62]: the hypervolume indicator determines the volume of the space in the objective space dominated by the front generated by a given method. Therefore, the HVR computes the proportion of the space dominated by the optimal Pareto front which is dominated by the approximation method.
  - IGD+, proposed by Ishibuchi et al. [29]: overcomes the drawbacks presented by GD [50] and IGD [16]. Following [10], this measure takes into account the dominance relation between the elements of the fronts to be compared when computing the Euclidean distance and it is weakly Pareto compliant. As it represents a distance between the fronts, a lower value is considered to be better.

- Distribution and spread:
  - Spread, proposed by Deb et al. [18]: takes into account the extent of the Pareto front approximation. A lower value is preferable. A spread value equal to 0 represents the most widely and evenly distributed set of non-dominated solutions.

6.3. First results using AUGMECON

We start by reporting on some results obtained when using the AUGMECON algorithm for finding the Pareto front of the problem we are investigating. The exact Pareto front could be obtained only for the J30 and J60 instances, although not for all. For the larger instances,

Table 10
Metrics and CPU time for the J120 instances such that AUGMECON could not successfully solve all the MILP problems.

	AUGMECON				Metaheuristic				
Instance	OVNG	C (F_1, F_2)	Γ	Time (hrs)	OVNG	M^*_3	Γ	ϵ	Time (hrs)
J1201_1	1	0%	—	124.05	45	—	—	—	6.07
J12010_1	1	30.3%	—	200	33	—	—	—	5.55
J12020_1	3	40.54%	0.748	140.08	37	6.652	1.143	0.667	10.21
J12030_1	1	30.56%	—	200	36	—	—	—	9.18
J12040_1	0	—	—	200	47	—	—	—	7.4
J12050_1	1	20%	—	200	20	—	—	—	6.97
J12060_1	1	16.13%	—	200	31	—	—	—	7.68
Average	1.1	–	–	180.6	35.6	–	–	–	7.6

AUGMECON could not find an exact single Pareto front.

In Table 3, we present the analysis for the instances with 30 and 60 activities. In this table, we detail the instances according to three project characteristics: Network complexity (NC), resource factor (RF), and resource strength (RS) which were described above. Observing Table 3 we realize that for 12 out of the 48 instances with 30 activities, the optimal Pareto front could not be obtained. For the J60 this number raises to 34.

To devise some possible relation between the difficulty in finding the optimal Pareto front and the specific features of the instances, we present Tables 4a and 4b, where a 1 indicates that for the corresponding instance, the optimal Pareto front could be obtained and a 0 indicates the contrary.

In these tables, we conclude for a clear tendency of AUGMECON to find the optimal Pareto fronts more easily when the resource factor decreases and when the resource strength increases. This is not surprising since a smaller RF indicates that the activities are not very resource-demanding which makes it 'easier' to allocate the resources. On the other hand, an increased resource strength makes it more difficult to process activities in parallel and thus makes it 'easier' to find a schedule for the activities.

In what follows, we use the instances for which the optimal Pareto front was obtained to benchmark the metaheuristic. Once this has been done and it becomes clear that the metaheuristic provides a good approximation of the Pareto front, we apply the heuristic to the other instances (the optimal Pareto front is not known) and analyze the results.

6.4. Heuristic benchmarking

The quality of the metaheuristic developed in Section 5 can be assessed by considering the instances for which AUGMECON could solve up to proven optimality all the MILP models called by the algorithm. For these instances we computed all the metrics described in Section 6.2.3.

To obtain reliable results, before calculating any metric, the objective values of each of the objective functions of the problem were normalized, according to the following formula:

$$f'_i(y) = (f_i(y) - \min_x f_i(x)) / (\max_x f_i(x) - \min_x f_i(x))$$

where $\max_x f_i(x)$ and $\min_x f_i(x)$ are the maximum and minimum values of the i-th objective function on the reference Pareto front, that is, on the front obtained by AUGMECON.

Table 5 shows all the metrics for the J30 instances such that AUGMECON could successfully solve all the MILP problems, as well as the CPU time (in hours) required by each method to solve the problems. Let us recall that for each instance in J30 and J60, AUGMECON has a time limit of 200 h (100 breakpoints, 2 h per breakpoint) and the metaheuristic a limit of 20 million of evaluations. As described in Section 6.2.3, OVNG gives the number of non-dominated points on the front. We can observe that, in most instances with 30 activities, OVNG is similar for both methods. Specifically, the average OVNG for AUGMECON was around 39 and for the metaheuristic around 36. $C(F1, F2)$ measures the fraction of solutions in the approximate Pareto front that are dominated by one or more solutions on the front provided by AUGMECON. We see that there are 5 instances where this metric is less than 100% which means that some of the solutions of the approximate front are not dominated by solutions on the exact front. $C(F2, F1)$, i.e., the fraction of solutions on the optimal Pareto front that are dominated by the approximate front, is not shown as it is always 0% in the instances of this table. Γ measures the size of the holes on the front, therefore, a lower value of this metric is desirable. The values for this metric are very similar in both methods, indicating that the maximum distance between adjacent points on both fronts is similar. Regarding M^*_3, it measures the extent of the front. Note that, due to normalization, the M^*_3-metric for the AUGMECON method is always equal to $\sqrt{2} \approx 1.414$ and for this reason, it is not shown in the table. In the approximate front, the minimum value for the M^*_3-metric is 1.193 and the maximum value is 1.528, therefore, the extent of the approximate Pareto front is not very different from the optimal Pareto front. The ϵ indicator gives the minimum additive factor by which the approximation set has to be translated in the objective space in order to (weakly) dominate the reference set. The values of this metric are, as we can observe in the table, quite low, which is desirable. Regarding HVR, we can observe that in 30 of the 36 instances it is greater than 90%. Let us recall that HVR computes the proportion of the space dominated by the optimal Pareto front which is dominated by the approximation method. The last two metrics, IGD+ and Spread, are mainly useful for comparing our metaheuristic with other methods in future research. IGD+ measures the distance between the set offered as an approximation to the Pareto front and the optimal Pareto front. We can see that IGD+ is quite low in all cases, which indicates that the approximate front is quite close to the optimal Pareto front. The spread takes into account the extent of the Pareto front approximation. A spread value equal to 0 represents the most widely and evenly distributed set of non-dominated solutions. In our case, these values are between 0.445 y 0.852.

In general, and taking into account all the metrics considered, we can conclude that the approximate Pareto fronts provided by the metaheuristic are quite good with respect to cardinality, distribution, spread and convergence. Moreover, the average CPU time to solve the instances in Table 5 was 3.9 h for AUGMECON and 0.6 h for the metaheuristic. Therefore, the results demonstrate that the metaheuristic is a good alternative to the exact method when solving these instances, which are the smallest of all those selected.

Table 6 shows the metrics for the J60 instances such that AUGMECON could successfully solve all the MILP problems, and the CPU time employed by both methods. For these 14 instances, AUGMECON gives fronts with, on average, 57 points in an average CPU time of 20 h per front. By contrast, the fronts obtained by the metaheuristic in 1.8 h, on average, have around 43 points. Regarding $C(F1, F2)$, we see that there is 1 instance where this metric is less than 100% which means that some of the solutions of the approximate front are not dominated by solutions on the exact front. Again, $C(F2, F1)$ is not shown, as it is always 0% in these instances. In most cases, the Γ values are very similar in both methods, i.e., the maximum distance between adjacent points in both fronts is similar. In the approximate front, the M^*_3-metric is between 0.971 and 1.587, compared with the value of $\sqrt{2}$ which AUGMECON always reports. Again, the values of the ϵ indicator are rather low, which is desirable. Regarding HVR, we can observe that in 11 of the 14 instances it is greater than 80%. IGD+ are quite low in all cases, indicating that the approximate front is close to the optimal Pareto front. Finally, we can see that the spread values are between 0.543 y 0.857. As in the analysis of the J30 instances, these results show that the metaheuristic provides approximations of the optimal Pareto fronts with good features in much lower computation times.

Fig. 6 graphically shows the Pareto fronts obtained, both by AUGMECON and by the metaheuristic, for two selected instances, one with 30 activities and the other with 60. As can be seen from the analysis of the previous metrics, in this figure we once again observe that the exact fronts and those obtained by the proposed metaheuristic are rather similar. Furthermore, we see that these similarities are stronger for shorter makespan, which will generally be the schedules in which a decision maker is usually more interested.

6.5. In search for approximate Pareto fronts

We now focus on the results for the instances for which AUGMECON could not find the exact Pareto front. Table 7 shows the metrics for the J30 instances such that AUGMECON could not successfully solve all the MILP problem, which happened in 12 of the 48 instances selected with 30 activities. The table also shows the CPU time (in hours) employed by

AUGMECON and the metaheuristic. In most of these instances, the metaheuristic was able to find more non-dominated solutions than AUGMECON. Specifically, the average OVNG for AUGMECON was around 27 and for the metaheuristic it was near 33. Regarding the C-metric, in most cases there are solutions on the front provided by the metaheuristic that are non-dominated by the solutions on the front provided by AUGMECON, that is, $C(F_1, F_2)$ is less than 100%. In cases where AUGMECON cannot find the optimal Pareto front, the fraction of solutions on the front provided by AUGMECON that are dominated by one or more solutions on the front provided by the metaheuristic, $C(F_2, F_1)$, may be nonzero. In the instances of Table 7, this happens in instances J3013_1, J3025_1, J3029_1, J3041_1 and J3045_1 where $C(F_2, F_1)$ was 4.55%, 62.5%, 5.26%, 25%, and 16.67%, respectively. We can see that the Γ values are very similar in both methods in most cases, which means that the maximum distance between adjacent points on both fronts is similar. The M^*_3-metric for the metaheuristic is greater than $\sqrt{2}$ in 7 of the 12 instances, that is, the extent of the metaheuristic front is greater than that of AUGMECON. Regarding the ϵ indicator, the values are rather low, which is desirable. Note that the metrics HVR, IGD+ and Spread are not calculated since we do not know the optimal Pareto front. Therefore, in these instances, the metaheuristic also obtains good fronts in 0.5 h on average in contrast to the more than 58 h needed by AUGMECON.

Table 8 details the metrics and the CPU time for the 34 instances with 60 activities such that AUGMECON could not successfully solve all the MILP problems. In most of the instances of Table 8, the metaheuristic was able to find more non-dominated solutions than AUGMECON, near 40 vs 22, on average. In four of the instances, the exact technique only obtains one point on the front in 200 h of execution time while the metaheuristic obtains for those instances, fronts with on average 50 points in 1.6 h, on average. Regarding the C-metric, in 21 of the 34 instances $C(F_1, F_2)$ is less than 100%. $C(F_2, F_1)$ was nonzero only in instance J605_1, with a value of 16.67%. In 16 of the instances, the Γ metric is better in the metaheuristic, in 13 instances this happens the other way around. Note that when the payoff table could not be approximated because one of the extreme points could not be calculated by AUGMECON, normalization could not be performed and, therefore, most of the metrics could not be calculated. This is the case for instances J6013_1, J6025_1, J6037_1 and J6045_1. The M^*_3-metric for the metaheuristic is greater than for AUGMECON in 20 of the 30 instances for which it can be calculated, that is, the extent of the metaheuristic front is better than that of AUGMECON. Regarding the ϵ indicator, the values are rather low again, which is desirable. For these instances, the average CPU time for the metaheuristic was 1.8 h, while for AUGMECON it grew to 88.1 h.

The results presented so far indicate that the metaheuristic proposed is a very good alternative to the exact method because it has demonstrated a good performance in the instances in which the exact method is able to give a front in extremely lower CPU times.

For the J90 and J120 instances selected, AUGMECON was run for a maximum of 200 h (50 breakpoints, 4 h per breakpoint) and it could not find the optimal Pareto front in any of the instances. Table 9 shows the metrics and CPU time for the J90 instances. In all the J90 instances, the metaheuristic was able to find more non-dominated solutions than AUGMECON. The average OVNG for AUGMECON was around 9 and near 42 for the metaheuristic. We can see that $C(F_1, F_2)$ is less than 100% in all cases. For example, in J9010_1, the 63.64% of the front given by the metaheuristic is not dominated by solutions on the AUGMECON front. Regarding the Γ and M^*_3 metrics, it is worth highlighting the case of the instance J9010_1 where the value for the metaheuristic is very high but, if we analyze in detail the fronts given by both methods (see Fig. 7c below), we can see that this is due to the little extent of the reference front provided by AUGMECON, which is also shown by the high value of the M^*_3 metric for the metaheuristic. The ϵ indicator, as always, takes quite good values. The average CPU time for the metaheuristic was 3.8 h, while for AUGMECON it was 118.5 h.

Table 10 details the metrics and CPU time for the J120 instances selected. As we can see, only in one of the instances with 120 activities, AUGMECON was able to find more than one point on the front, which allows the calculation of the analyzed metrics. What's more, for one of the instances (J12040_1) it was not able to find any point. For these instances, AUGMECON gives fronts with a number of points which varies from 0 to 3 in around 180 h of CPU time, on average, and the metaheuristic gives, in less than 8 h per instance, fronts with, on average, near 36 points. $C(F_1, F_2)$ is less than 50% in all cases. Regarding the high values of the Γ and M^*_3 metrics for instance J12020_1, these are explained again by the little extent of the reference front obtained by AUGMECON (see Fig. 7d below).

The last results show that the exact method performs as intractable to solve the problem considered in large instances. In these cases, the metaheuristic, which has demonstrated a good performance in small or medium sized instances, has nowadays become the only practical alternative and can find good approximate fronts in reasonable computation times.

Fig. 7 shows the Pareto fronts obtained, both by AUGMECON and by the metaheuristic, for 4 instances, one for each one of the sets in PSPLIB we have considered in this work. These are only four examples where the metaheuristic gives fronts with better characteristics than the exact method with regard to the different type of features considered for comparing the fronts: cardinality, convergence, distribution and spread.

7. Conclusions and future research

In this paper, we have presented a bi-criteria resource-constrained project scheduling problem considering as objective functions the makespan and the total cost associated with resource usage, which is time-dependent. Several major conclusions can be drawn from the work carried out. First, only for small to medium sized instances was it possible to find exact Pareto solutions. Still, in many cases, the computational effort required is significant. Second, the problem we are investigating is quite rich in terms of the Pareto solutions found. This means that, in general, each instance of the problem leads to a large set of Pareto solutions. Third, the metaheuristic developed for the problem is quite effective in finding the approximate Pareto front. We could observe results that often correspond to sharp approximations of the Pareto front. Furthermore, the CPU time required by the metaheuristic is rather small given the quality of the solutions found. Moreover, when the use of AUGMECON, even as an approximate method failed to find Pareto solutions to the problem, the metaheuristic was able to deliver a rich set of approximate solutions in reasonable computation times. Overall, finding a perfect cost-time trade-off for the RCPSP is far from possible since many compromise solutions can be adopted. In any case, the methodologies proposed in this paper make it possible to provide a decision maker with a rich set of alternative solutions from which a better decision can certainly be made.

Several research avenues are opened with this work. In fact, the set of objective functions investigated in this work can be extended to consider other possibilities such as resource leveling to ensure an even use of the resources throughout the planning horizon. The use of non-renewable resources is also an interesting research direction to explore. Above all, multicriteria resource-constrained project scheduling problems define a very challenging area in which much work still remains to be done. The use of time dependent costs for the resources is an interesting area that is still very much unexplored.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors thank the grants PID2019-105952GB-I00 funded by Ministerio de Ciencia e Innovación/ Agencia Estatal de Investigación /10.13039/501100011033, Spain, PROMETEO/2021/063 funded by the government of the Valencian Community, Spain and UIDB/04561/2020 by National Funding from FCT—Fundação para a Ciência e Tecnologia, Portugal.

References

[1] Abbasi B, Shadrokh S, Arkat J. Bi-objective resource-constrained project scheduling with robustness and makespan criteria. Appl Math Comput 2006;180:146–52.
[2] Abdolshah M. A review of resource-constrained project scheduling problems (RCPSP) Approaches and solutions. Int Trans J Eng, Manag, Appl Sci Technol 2014;5:253–86.
[3] Abello MB, Michalewicz Z. Multiobjective resource-constrained project scheduling with a time-varying number of tasks. Sci World J, Artic 2014:420101.
[4] Achuthan NR, Hardjawidjaja A. Project scheduling under time dependent costs–a branch and bound algorithm. Ann Oper Res 2001;108(1–4):55–74.
[5] Al-Fawzan M, Haouari M. A bi-objective model for robust resource-constrained project scheduling. Int J Prod Econ 2005;96:175–87.
[6] Alcaraz J, Maroto C. A robust genetic algorithm for resource allocation in project scheduling. Ann Oper Res 2001;102(1–4):83–109.
[7] Alcaraz J, Maroto C. A hybrid genetic algorithm based on intelligent encoding for project scheduling. In: Józefowska J, Weglarz J, editors. Perspectives in Modern Project Scheduling, volume 92 of International Series in Operations Research and. Springer; 2006. p. 250–74.
[8] Alcaraz J, Maroto C, Ruiz R. Solving the multi-mode resource-constrained project scheduling problem with genetic algorithms. J Oper Res Soc 2003;54(6):614–26.
[9] Artigues C, Koné O, Lopez P, Mongeau M. The resource-constrained project scheduling problem: mixed-integer linear programming formulations. In: Schwindt C, Zimmermann J, editors. Handbook on project management and scheduling. Cham: Springer International Publishing; 2015. p. 17–41.
[10] Audet C, Bigeon J, Cartier D, Digabel SL, Salomon L. Performance indicators in multiobjective optimization. Eur J Oper Res 2021;292:397–422.
[11] Ballestín F, Blanco R. Multi-criteria objectives in project scheduling: theoretical and practical fundamentals. In: Schwindt C, Zimmermann J, editors. Handbook on project management and scheduling. Cham: Springer International Publishing; 2015. p. 411–27.
[12] Boctor F. Resource-constrained project scheduling by simulated annealing. Int J Prod Res 1996;34:2335–51.
[13] Boland N, Charkhgard H, Savelsbergh M. A criterion space search algorithm for biobjective integer programming: the balanced box method. INFORMS J Comput 2015;27:735–54.
[14] Cervantes M, Lova A, Tormos P, Barber F. A dynamic population steady-state genetic algorithm for the resource-constrained project scheduling problem. Applied Artificial Intelligence. Berlin, Heidelberg: Springer; 2008. p. 611–20. . In N.T., N., L., B., A., G., and M., A., editors.
[15] Chaleshtarti AS, Shadrokh S, Khakifirooz M, Fathi M, Pardalos PM. A hybrid genetic and lagrangian relaxation algorithm for resource-constrained project scheduling under nonrenewable resources. Appl Soft Comput J 2020:94.
[16] Coello C, Cortés C. Solving multiobjective optimization problems using an artificial immune system. Genet Program Evol Mach 2005;6:163–90.
[17] Custòdio AL, Madeira J, Vaz A, Vicente L. Direct multisearch for multiobjective optimization. SIAM J Optim 2011;21:1109–40.
[18] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multi-objec- tive genetic algorithm: NSGA-II. IEEE Trans Evolut Comput 2002;6:182–97.
[19] Durillo JJ, Nebro AJ. jMetal: A Java framework for multi-objective optimization. Adv Eng Softw 2011;42:760–71.
[20] Fang C, Wang L. An effective shuffled frog-leaping algorithm for resource-constrained project scheduling problem. Comput Oper Res 2012;39(5):890–901.
[21] Fernandes A, Rodrigues C, daCosta FA. A path-relinking algorithm for the multi-mode resource-constrained project scheduling problem. Comput Oper Res 2018;92:145–54.
[22] Florez L, Castro-Lacouture D, Medaglia AL. Sustainable workforce scheduling in construction program management. J Oper Res Soc 2013;64:1169–81.
[23] Habibi F, Barzinpour F, Sadjadi S. Resource-constrained project scheduling problem: review of past and recent developments. J Proj Manag 2018;3:55–88.
[24] Habibi F, Barzinpour F, Sadjadi SJ. A multi-objective optimization model for project scheduling with time-varying resource requirements and capacities. J Ind Syst Eng 2017;10:92–118.
[25] Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling. Nav Res Logist 1998;45:733–50.
[26] Hartmann, S. (2006). Project Scheduling under Limited Resources. Models, Methods, and Applications, volume 478 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag Berlin Heidelberg.
[27] Hartmann S, Briskorn D. A survey of variants and extensions of the resource-constrained project scheduling problem. Eur J Oper Res 2010;207:1–14.
[28] Hartmann S, Briskorn D. An updated survey of variants and extensions of the resource-constrained project scheduling problem. Eur J Oper Res 2022;297(1):1–14.
[29] Ishibuchi H, Masuda H, Tanigaki Y, Nojima Y. Modified distance calculation in generational distance and inverted generational distance. In: Gaspar-Cunha A, Antunes CH, Coello C, editors. Evolutionary multi-criterion optimization. Springer; 2015. p. 110–25.
[30] Kelley, J. and Walker, M. (1959). Critical-path planning and scheduling. In Heart, F., editor, IRE-AIEE-ACM '59 (Eastern): Papers presented at the December 1–3, 1959, eastern joint IRE-AIEE-ACM computer conference. Association for Computing Machinery, New York, NY, United States.
[31] Kolisch R, Hartmann S. Heuristic algorithms for solving the resource constrained project scheduling problem: classification and computational analysis. In: Weglarz J, editor. Project scheduling: recent models, algorithms and applications. Kluwer; 1999. p. 147–78.
[32] Kolisch R, Sprecher A. PSPLIB - A project scheduling problem library: OR software - ORSEP operations research software exchange program. Eur J Oper Res 1997;96:205–16.
[33] Lin Q, Li J, Du Z, Chen J, Ming Z. A novel multi-objective particle swarm optimization with multiple search strategies. Eur J Oper Res 2015;247(3):732–44.
[34] Liu Y, Dong H, Lohse N, Petrovic S, Gindy N. An investigation into minimising total energy consumption and total weighted tardiness in job shops. J Clean Prod 2014;65:87–96.
[35] Martins P. Integrating financial planning, loaning strategies and project scheduling on a discrete-time model. J Manuf Syst 2017;44:217–29.
[36] Mavrotas G. Effective implementation of the ε -constraint method in multi-objective mathematical programming problems. Appl Math Comput 2009;213:455–65.
[37] Mejía G, Pereira J. Multiobjective scheduling algorithm for flexible manufacturing systems with petri nets. J Manuf Syst 2020;54:272–84.
[38] Möhring RH, Schul AS, Stork F, Uet M. Solving project scheduling problems by minimum cut computations. Manag Sci 2003;49:330–50.
[39] Moon J-Y, Park J. Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage. Int J Prod Res 2014;52(13):3922–39.
[40] Nabipoor Afruzi E, Najafi AA, Roghanian E, Mazinani M. A multi-objective imperialist competitive algorithm for solving discrete time, cost and quality trade-off problems with mode-identity and resource-constrained situations. Comput Oper Res 2014;50:80–96.
[41] Nebro AJ, Durillo JJ, Vergne M. Redesigning the jMetal multi-objective optimization framework. Proceedings of the companion publication of the 2015 annual conference on genetic and evolutionary computation. New York, NY, USA: Association for Computing Machinery; 2015. p. 1093–100.
[42] Pellerin R, Perrier N, Berthaut F. A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. Eur J Oper Res 2020;280:395–416.
[43] Pottel S, Goel A. Scheduling activities with time-dependent durations and resource consumptions. Eur J Oper Res 2022;301(2):445–57.
[44] Pritsker A, Watters L, Wolfe P. Multi-project scheduling with limited resources: a zero-one programming approach. Manag Sci 1969;16:93–108.
[45] Sato H, Aguirre HE, Tanaka K. Self-controlling dominance area of solutions in evolutionary many-objective optimization. In: Deb K, Bhattacharya A, Chakraborti N, Chakroborty P, Das S, Dutta J, et al., editors. Simulated Evolution and learning. Berlin, Heidelberg: Springer; 2010. p. 455–65 (Berlin Heidelberg).
[46] Serrano-Ruiz JC, Mula J, Poler R. Smart manufacturing scheduling: a literature review. J Manuf Syst 2021;61:265–87.
[47] Srinivas N, Deb K. Multiobjective function optimization using nondominated sorting genetic algorithms. Evolut Comput 1995;2:221–48.
[48] Szmerekovsky JG, Venkateshan P. An integer programming formulation for the project scheduling problem with irregular time-cost tradeoffs. Comput Oper Res 2012;39(7):1402–10.
[49] Ulusoy G, Serifoglu F, Sahin S. Four payment models for the multi-mode resource constrained project scheduling problem with discounted cash flows. Ann Oper Res 2001;102:237–61.
[50] van Veldhuizen, D.A., Lamont, G.B. (1999). Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, School of Engineering of the Air Force Institute of Technology, Dayton, Ohio.
[51] Wang J, Hu X, Demeulemeester E, Zhao Y. A bi-objective robust resource allocation model for the RCPSP considering resource transfer costs. Int J Prod Res 2021;59:367–87.
[52] Wang, X., Dugardin, F., and Yalaoui, F. (2016). An exact method to solve a bi-objective resource constraint project scheduling problem. IFAC-PapersOnLine, 49:

1038–1043.8th IFAC Conference on Manufacturing Modelling, Management and Control MIM 2016.
[53] Wang, X., Yalaoui, F., and Dugardin, F. (2017). Genetic algorithms hybridized with the self controlling dominance to solve a multi-objective resource constraint project scheduling problem.In 2017 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI), 39–44.
[54] Wang X, Yalaoui F, Dugardin F. Non-dominated sorting genetic algorithms for a multi-objective resource constraint project scheduling problem. J Intell Syst 2019; 28:791–806.
[55] Wu C-C, Bai D, Zhang X, Cheng S-R, Lin J-C, Wu Z-L, et al. A robust customer order scheduling problem along with scenario-dependent component processing times and due dates. J Manuf Syst 2021;58:291–305.
[56] Yuan Y, Xu H, Wang B, Yao X. A new dominance relation-based evolutionary algorithm for many-objective optimization. IEEE Trans Evolut Comput 2016;20(1): 16–37.
[57] Yuan Y, Xu H, Wang B, Zhang B, Yao X. Balancing convergence and diversity in decomposition-based many-objective optimizers. IEEE Trans Evolut Comput 2016; 20(2):180–98.
[58] Zamani R. An effective mirror-based genetic algorithm for scheduling multi-mode resource constrained projects. Comput Ind Eng 2019;127:914–24.
[59] Zheng X, Wang L. A multi-agent optimization algorithm for resource constrained project scheduling problem. Expert Syst Appl 2015;42(15):6039–49.
[60] Zitler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolut Comput 2000;8:173–95.
[61] Zitler E, Thiele L. Multiobjective optimization using evolutionary algorithms–a comparative case study. In et al., A. E., editor, Parallel Problem Solving from Nature. Springer; 1998. p. 292–301.
[62] Zitzler, E. (1999). Evolutionary algorithms for multiobjective optimization: Methods and applications. PhD thesis, Swiss Federal Institute of Technology.
[63] Zitzler E, Thiele L, Laumanns M, Fonseca CM, daFonseca VG. Performance assessment of multiobjective optimizers: an analysis and review. IEEE Trans Evolut Comput 2003;7:117–32.