Computers & Industrial Engineering 162 (2021) 107592

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A matheuristic approach to the multi-mode resource constrained project scheduling problem

Gustavo Alves Fernandes, Sérgio Ricardo de Souza *

Federal Center of Technological Education of Minas Gerais (CEFET-MG), ZIP Code: 30510-000, Belo Horizonte, MG, Brazil

ARTICLE INFO

Keywords:
Multi-mode resource-constrained project scheduling problem
Matheuristic
Local branching
Project scheduling

ABSTRACT

This article addresses the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP). It is an NP-Hard problem and consists of scheduling all activities of a project, respecting the precedence order of these activities, which, once initiated, cannot be interrupted. Jobs have combinations of execution modes, in the form of different combinations of processing time and resource consumption. The objective is to minimize the duration of the project, finding start times for each project activity. This article proposes a Local Branching matheuristic strategy for solving MRCPSP. The influence of the parameters of the based local search method is verified, as well as the characterization of neighborhood structures that define a subproblem during the enumeration of the LB matheuristic. Two characterizations to define the subproblems are proposed, and one of them surpasses the characterization discussed in the literature. Computational tests were performed using sets of instances from the PSPLIB and MMLIB libraries. The best-known results for these instances in relation to the state-of-the-art are presented as a reference to validate the results obtained by the approach proposed in this article.

1. Introduction

In project management, it is fundamental to define which resources will be used in the execution, the availability of each one of them, and the consumption estimation of these resources by the executed activities. A widely used project resources classification presented in Blazewicz (1986) distinguishes them in (i) renewable resources; (ii) nonrenewable resources; and (iii) doubly restricted resources. Renewable resources are available from period to period of the project, meaning their total value is renewed each period as the activities are completed. Typical examples of renewable resources include labor, machinery, tools, equipment, and space. Non-renewable resources, on the other hand, are not returned to be available again once used by an activity. A simple example of such resources is the capital associated with the project. Other typical examples are raw materials and fuels. Doubly restricted resources are restricted by a period as well as throughout the project horizon. An example of this case is worker-hour per day, where the total number of workers is available daily; however, there is a restriction of the total number of hours worked per day for each worker.

Demeulemeester (2002) highlights that several issues related to the context of scheduling and production operations share characteristics that make them susceptible to techniques developed in the context of project scheduling. Demeulemeester (2002) further adds that several studied scheduling problems, such as Single Machine Problems, Parallel Machine Problems, Flow Shops, and Job Shops, are modeled as a classical problem known as Resource-Constrained Project Scheduling Problem (RCPSP). RCPSP consists of scheduling the activities, respecting the precedence between them, as well as the consumption of renewable resources. Hartmann and Briskorn (2010) discuss variations of the RCPSP, such as in objectives, resources, or activities, with the justification that the original RCPSP does not cover all situations that may occur in practice.

The current article addresses a generalization of RCPSP known as Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) and presents a variation of the Local Branching (LB) method, proposed by Fischetti and Lodi (2003), as a matheuristic for the resolution of MRCPSP. The LB strategy resembles the known metaheuristics of local search, but, unlike these, the neighborhoods are obtained by pseudo-linear inequalities known as local branching cuts. These pseudo-inequalities are inserted into a Mixed-Integer Programming (MIP) to ensure local optimality. Some recent approaches based on LB for solving integer optimization problems are, among others, Acuna-Agost, Michelon, Feillet, and Gueye (2011) for rail sequencing, Rodríguez-

* Corresponding author.
E-mail addresses: gustavo.alfer@gmail.com (G.A. Fernandes), sergio@dppg.cefetmg.br (S.R. de Souza).

https://doi.org/10.1016/j.cie.2021.107592
Received 20 January 2021; Received in revised form 23 June 2021; Accepted 28 July 2021
Available online 9 August 2021

Martín and Salazar-González (2010) for network flow problems, Li, Bookbinder, and Elhedhli (2012) for load planning and Zhu, Bard, and Yu (2006) for MRCPSP.

We propose a matheuristic scheme based on a Branch-and-Cut (B&C) algorithm, which uses information from MRCPSP to characterize the subproblems (or neighborhoods), by a pseudo-linear inequality, during the search of B&C. Thus, this article verifies the influence of four pseudo-linear inequality classes, which are added to the B&C strategy, during the search of the solution space. By adding an inequality, it generates a subproblem which must be solved. Then, each inequality class has a quality associated with the generated bound as a subproblem is solved. The proposal presented here consists of solving a MIP to find an initial solution $\bar{x}$, based on the shortest duration of the activities. This solution is refined by a local search procedure and is used by the LB strategy, which, for each neighborhood generated around $\bar{x}$, searches for local optimality. For each new local optimal solution x_L^* found, the region around solution $\bar{x}$ is deleted, and the search resumes around x_L^*. The main contribution of this article is the characterization of two classes of neighborhoods for MRCPSP as pseudo-cuts for the local branching cut variant since this method is dependent on the subproblem characterization.

This article is organized as follows. Section 2, in the sequence, characterizes the problem. Section 3 shows a literature review concerning the solution of MRCPSP. In the following, Section 4 addresses the adaptation of the LB strategy for solving MRCPSP. Section 5 presents the computational results concerning the application of the proposed procedure. Section 6 ends the article, showing conclusions about the described proposal.

2. Problem description

The Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP) is characterized by activities $j = 1, \ldots, n$ of a project, that must be processed without interruption and subject to the relation of precedence between them. Thus, each activity j must be started only after its predecessor i is completed, for $(i,j) \in \mathcal{P}$, where $\mathcal{P}$ is the set of precedence constraints. Let $\mathcal{S}$ be the set of successors, such that $(i,j) \in \mathcal{S}$ means that the activity j is a successor of i. By convention, $j = 1$ and $j = n$ are dummy activities, such that $j = 1$ and $j = n$ should be, respectively, the first and last activities in the schedule. Each activity j is performed in an execution mode $m \in M_j$, which establishes the processing time d_{jm}, the demand for renewable resources r^{ρ}_{jmk} and for nonrenewable resources r^{ν}_{jmk} of type k, for $k = 1, \ldots, |K|$. Renewable resources are replenished at their maximum capacity R^{ρ}_k at each period t. T is an upper bound for the total project makespan. The nonrenewable resources have an available capacity R^{ν}_k for the entire duration of the project and are not replenished. All parameters are non-negative integers.

The goal for MRCPSP is to find a feasible combination of the execution modes for all activities, as well as to determine a feasible schedule subject to resource and precedence constraints, such that the project duration, or makespan, is minimal. MRCPSP is an NP-Hard problem, according to Blazewicz, Lenstra, and Kan (1983). Kolisch and Drexl (1997) showed that finding a feasible solution for MRCPSP with at least two types of nonrenewable resources and at least two modes of execution is an NP-Hard problem.

Let x_{jmt} be the decision variable so that $x_{jmt} = 1$ if the activity j is executed in mode m and finishes in period t. Otherwise, $x_{jmt} = 0$. The mathematical formulation for MRCPSP, introduced by Talbot (1982), is given by:

$$(P) \quad \min \sum_{m=1}^{|M_n|} \sum_{t=EFT_n}^{LFT_n} t x_{nmt} \tag{1}$$

s.t.

$$\sum_{m=1}^{|M_j|} \sum_{t=EFT_j}^{LFT_j} x_{jmt} = 1, \quad j = 1, \ldots, n \tag{2}$$

$$\sum_{m=1}^{|M_i|} \sum_{t=EFT_i}^{LFT_i} t x_{imt} \leqslant \sum_{m=1}^{|M_j|} \sum_{t=EFT_j}^{LFT_j} (t - d_{jm}) x_{jmt}, \quad \forall (i,j) \in \mathcal{P} \tag{3}$$

$$\sum_{j=1}^{n} \sum_{m=1}^{|M_j|} \sum_{s=\max\{t, EFT_j\}}^{\min\{t+d_{jm}-1, LFT_j\}} r^{\rho}_{jmk} x_{jms} \leqslant R^{\rho}_k, \quad k = 1, \ldots, R^{\rho}; t = 1, \ldots, T \tag{4}$$

$$\sum_{j=1}^{n} \sum_{m=1}^{|M_j|} \sum_{t=EFT_j}^{LFT_j} r^{\nu}_{jmk} x_{jmt} \leqslant R^{\nu}_k, k = 1, \ldots, R^{\nu} \tag{5}$$

$$x_{jmt} \in \{0,1\}, j = 1, \ldots, n; m = 1, \ldots, |M_j|; t = EFT_j, \ldots, LFT_j \tag{6}$$

For (P) formulation, Expression (1) shows the objective of minimizing the completion time of the last activity. Expression (2) represents the execution of one mode per activity, as well as the assignment of only one finish time. Expression (3) shows the precedence relation between activities. Expression (4) ensures the availability of renewable resources by period is not violated. Expression (5) corresponds to the restriction of nonrenewable resources. Finally, Expression (6) shows the binary domain of the decision variable x_{jmt}.

The values EST_j and EFT_j define, respectively, the Earliest Start Time and the Earliest Finish Time of activity j. They are obtained by the Critical Path Method, as shown in Kelley and Walker (1959), after the assignment of the shortest duration feasible mode for each activity. The values LST_j and LFT_j, which define the Latest Start Time and the Latest Finish Time of activity j, respectively, are found by assigning the largest duration feasible mode to each activity.

3. Literature review

This section presents a literature review involving proposals for solving MRCPSP. Section 3.1 discusses articles that deal with solutions using mathematical programming methods. Section 3.2, in the sequel, discusses articles introducing solutions using metaheuristics. Weglarz, Józefowska, Mika, and Waligóra (2011) and Noori and Taghizadeh (2018) show a further complete overview concerning approaches associated with this problem. Additionally, Van Peteghem and Vanhoucke (2014) present a detailed review and analysis of heuristic methods, and Changchun, Xi, Canrong, Qiang, and Li (2018), in its turn, includes a listing of articles produced about MRCPSP.

3.1. Solutions using exact methods

Patterson, Slowinski, Talbot, and Weglarz (1989) proposed a branch-and-bound algorithm (B&B), named as the Precedence tree, whose schedule is built through an enumeration tree on the activity-mode pair. Sprecher (1994) and Sprecher and Drexl (1998) improved this approach, including other branching criteria. Sprecher, Hartmann, and Drexl (1997) presented a new B&B approach, introducing the concept of Mode and Delay Alternatives. This approach also builds a schedule on a tree. Later, using the concept of Mode and Delay Alternatives, Hartmann and Drexl (1998) created the concept of Mode and Extension Alternatives. These three B&B approaches can be found in detail with comparisons of each other in Hartmann and Drexl (1998).

Another exact approach to solving MRCPSP with minimization of makespan is due to Zhu et al. (2006) in which a Branch-and-Cut algorithm is presented for MRCPSP. The authors assert that the used approach consumes less runtime to determine an optimal schedule relative to the work of Sprecher and Drexl (1998). However, by evaluating the difference

in the computational effort of both approaches, the latter presents a faster enumeration. It should be highlighted that Zhu et al. (2006) found optimal values for 506 out of 552 instances of the largest class (J30) of the public library PSPLIB (Kolisch & Sprecher, 1997).

Schnell and Hartl (2017) analyzed exact methods involving a hybrid approach (CP-SAT) combining Constraint Programming (CP) and Boolean Satisfaction Resolution (SAT) and MIP techniques to solve the RCPSP, and proposed extensions of this hybrid approach to the MRCPSP solution. These authors were the first to report results involving mathematical programming for the MMLIB library (Van Peteghem & Vanhoucke, 2014), including providing optimal values for several instances of this library. Changchun et al. (2018) used a column-based approach to MRCPSP in a manufacturing supply chain context in which different product types are independently produced and coordinated by a third party logistics manager or resource manager who provides different types of vehicles to provide products and attend to the customer demand.

Gerhards, Stuerck, and Fink (2017) feature a matheuristic, which combines the Adaptive Large Neighborhood Search (ALNS) metaheuristic with Mixed Integer Programming. The proposed mathematical technique was used to solve instances of the MMLIB and MMLIB+ sets. The results show that the proposal is strongly competitive against the best literature results for these sets of instances.

Almeida, Correia, and Saldanha-da-Gama (2019) discussed a close variant of MRCPSP, known as Multi-skill Resource-constrained Project Scheduling Problem (MSRCPSP). Two mathematical models (an existing continuous-time formulation and an existing discrete-time formulation) from literature are revised and two new discrete-time formulations models are introduced for the problem. A theoretical comparison between the lower bounds of the linear programming relaxations are performed and shows that these lower bounds are never worse than the lower bounds provided by the other models.

3.2. Solutions using metaheuristics

Bouleimen and Lecocq (2003) show an implementation of the Simulated Annealing (SA) metaheuristic as a two-stage local search. In the first stage, the neighborhood is based only on execution modes, and, in cases where it is possible to improve makespan, the second stage is executed only in activity-based neighborhoods. Zhang, Tam, and Li (2006) discuss an application of the Particle Swarm Optimization (PSO) evolutionary metaheuristic with the addition of a procedure to repair infeasible solutions. Jarboui, Damak, Siarry, and Rebai (2008) use also the PSO metaheuristic to assign execution modes to activities, but, in this case, a local search procedure determines how to schedule the set of activities. In Mika, Waligora, and Weglarz (2008), an implementation of the Tabu Search (TS) metaheuristic is proposed for solving MRCPSP, considering setup times. An approach based on the Differential Evolution (DE) metaheuristic is presented by Damak, Jarboui, Siarry, and Loukil (2009), where the authors compare the found results concerning the results of Jarboui et al. (2008) and Bouleimen and Lecocq (2003). An Artificial Immune System (AIS) approach is used by Van Peteghem and Vanhoucke (2009), which employs the clonal selection principle to explore various regions of the search space.

Lova et al. (2009) present an implementation of a Genetic Algorithm (GA) hybridized with the Multi-mode Forward/Backward Improvement (MFBI) heuristic. Van Peteghem and Vanhoucke (2010) use the same hybridization technique; however, in this case, the authors highlight the contribution of MFBI to the improvement of GA performance. Okada et al. (2014) also present a GA hybridized with the MFBI and local search procedures based on the Critical Path. The Scatter-search method is implemented by Van Peteghem and Vanhoucke (2011), which also uses a heuristic in an attempt to construct feasible solutions. A procedure based on the Ant Colony Algorithm is used by Chiang, Huang, and Wang (2008) and Li and Zhang (2013). Li and Zhang (2013) employ Priority Rules to guide the ants in the choice of activities while building the schedule.

In Muller (2011), the Adaptive Large Neighborhood Search (ALNS) metaheuristic is used. This metaheuristic uses several neighborhood structures, and, during the search, the neighborhoods compete with each other to improve the current solution. Wauters, Verbeeck, Berghe, and De Causmaecker (2011) present a multi-agent approach, in which heuristics struggle to present the best schedule. Fernandes and de Souza (2017) introduce a hybridized strategy between AIS and ALNS metaheuristics, obtaining significant results for the PSPLIB instances.

Van Peteghem and Vanhoucke (2014) perform an extensive comparison between metaheuristics for MRCPSP resolution. They compare these metaheuristics using two well-known benchmark datasets, the PSPLIB dataset (Kolisch, Sprecher, & Drexl, 1995; Kolisch & Sprecher, 1997) and the Boctor dataset (Boctor, 1993). Finally, by detecting the weaknesses of these two benchmark datasets in the face of advances in solution procedures, it proposes two new sets of instances, called MMLIB and MMLIB+, that prove challenging to the current state of the art MRCPSP solution procedures.

Geiger (2017) presents a parallel hybrid algorithm involving Variable Neighborhood Search (VNS) and Iterated Local Search (ILS). The article presents results for the multi-mode, resource-constrained multi-project scheduling problems (MMRCPSP) and the multi-mode, resource-constrained project scheduling problems (MRCPSP). Concerning the MMRCPSP, the study refers to the instances used for the MISTA 2013 Challenge (Wauters et al., 2016). In the case of MRCPSP, our object of interest, the study shows quality results for the instances MMLIB and MMLIB+, which are strongly competitive with the best results known in the literature.

Muritiba, Rodrigues, and da Costa (2018) presented an approach based on the Path Relinking (PR) procedure. This implementation differs from other PR procedures in general in that it does not act as a post-intensification heuristic, but, in fact, constitutes the procedure metaheuristic itself. The computational results show that this proposal presents better heuristic results than other works of the literature so far, both concerning the PSLIB and Boctor datasets, as well as to the new instances sets MMLIB and MMLIB+.

Chakrabortty, Abbasi, and Ryan (2020) also present an approach based on VNS metaheuristic combined with MRCPSP specific heuristics known as priority rules to generate schedules as well as neighborhood structures. The results presented for the PSPLIB do not exceed those presented by Muritiba et al. (2018). Regarding the MMLIB library, the

Table 1
Summary of the bibliographical review, highlighting the methods used for solving the MRCPSP.

Author	Method	Local Search
Patterson et al. (1989)	B&B	–
Sprecher (1994)	B&B	–
Sprecher et al. (1997)	B&B	–
Sprecher and Drexl (1998)	B&B	–
Hartmann and Drexl (1998)	B&B	–
Bouleimen and Lecocq (2003)	SA	Yes
Zhu et al. (2006)	B&C	–
Zhang et al. (2006)	PSO	Yes
Jarboui et al. (2008)	PSO	Yes
Mika et al. (2008)	TS	Yes
Chiang et al. (2008)	ACO	–
Damak et al. (2009)	DE	–
Van Peteghem and Vanhoucke (2009)	AIS	Yes
Lova et al. (2009)	GA-MFBI	Yes
Van Peteghem and Vanhoucke (2010)	GA-MFBI	Yes
Van Peteghem and Vanhoucke (2011)	SS	Yes
Muller (2011)	ALNS	Yes
Wauters et al. (2011)	Multi-agent	Yes
Li and Zhang (2013)	ACO	–
Okada et al. (2014)	GA-MFBI	Yes
Schnell and Hartl (2017)	CP-SAT	–
Gerhards et al. (2017)	ALNS-MIP	–
Fernandes and de Souza (2017)	AIS-ALNS	Yes
Geiger (2017)	VNS-ILS	Yes
Muritiba et al. (2018)	PR	Yes
Chakrabortty et al. (2020)	VNS	Yes

results from Muritiba et al. (2018) are better for the problem classes MMLIB50 and MMLIB100, while for the MMLIB+ class, Chakrabortty et al. (2020) present the best results by the time of writing this paper.

Table 1 presents a summary of the literature review, highlighting the methods used to solve the MRP in each of the articles included, both for solutions via exact methods and for solutions via metaheuristics. Hartmann and Briskorn (2021) presents an updated survey of variants and extensions of the resource-constrained project scheduling problem, including the MRCPSP.

4. Proposal of a matheuristic for solving MRCPSP

This section presents the proposal for solving MRCPSP using matheuristics. First, Section 4.1 shows a brief review of matheuristics and its basic concepts. Then, Section 4.2 reviews the Local Branching procedure. Section 4.3 details the proposed algorithm as a matheuristic approach for solving MRCPSP.

4.1. A review of matheuristics

The literature review included in Section 3.1 presents several mathematical programming techniques proposed for the MRCPSP solution. However, according to Lodi et al. (2009, chap. 16), using these approaches purely is not the best way to solve scheduling problems. For Lodi et al. (2009, chap. 16), a challenging question is whether there is a good Integer Linear Programming (ILP) model for a scheduling problem, or, on the other hand, whether researchers should attempt to extend the problem model to different directions, such as hybridization.

Also according to the exposed in Section 3.1, Zhu et al. (2006), claimed as the state-of-the-art for the PSPLIB instances, is based on a Branch-and-Cut scheme, but, in its turn, also makes use of hybridization with a genetic algorithm. This methodology uses the MRCPSP mathematical formulation, and, at the same time, searches for heuristic and metaheuristic techniques that also explores the characteristics of this problem.

The Branch-and-Cut method (B&C) is a variation of the Branch-and-Bound (B&B) method (Bertsimas & Tsitsiklis, 1997; Mitchell, 2009), which uses cuts (linear inequalities) to solve the subproblems. The number of constraints of the subproblems is increased with additional linear inequalities, in order to improve the bounds obtained by some efficient algorithm. For Lodi et al. (2009, chap. 16), approaches based on cut generation are continually used, and the main discussion is about how to choose the cuts generated since the performance of a cut generation approach depends on the quality of the added inequality.

Matheuristics, as stated in Boschetti, Maniezzo, Roffilli, and Bolufé Röhler (2009), are currently receiving great attention in the literature. The objective of these procedures is to structure an algorithm for solving efficiently an optimization problem by combining both mathematical programming and (meta) heuristics methods. Matheuristics are described as hybrid algorithms between (meta) heuristics and mathematical programming. This hybridization occurs in two categories, according to Raidl and Puchinger (2008): *(i)* the (meta) heuristic controls the calls to the mathematical programming procedure; or *(ii)* the mathematical programming procedure is characterized as *master*, controlling the use of the (meta) heuristic.

In this article, we propose an investigation of a matheuristic procedure, performed according to category *(i)*. This means that the mathematical programming procedure is controlled by a heuristic, i.e., the mathematical programming procedure is carried out as a B&C scheme. A neighborhood structure will be characterized to generate the subproblems, and a pseudo-linear inequality, related to this neighborhood structure, will be used for subproblems generation.

4.2. Local branching: a review

This subsection presents the definitions of the Local Branching method. First we present the pseudo cut definition, afterwards we discuss the strategy enumeration scheme.

Let (IP) be a mixed-integer programming problem (MIP):

$$(IP) \quad \min \quad c'x \tag{7}$$

$$\text{s.t.} \quad Ax \geqslant b \tag{8}$$

$$x \geqslant 0, \tag{9}$$

$$x_j \in \{0,1\}, j \in \mathbb{N}_+ \tag{10}$$

and let $\overline{x}$ be a reference feasible solution associated to (IP).

Fischetti and Lodi (2003) introduced the Local Branching (LB) method for the solution of MIPs, based on the settings of variable values to guarantee the optimality in the neighborhood of $\overline{x}$. This paper also presented how to define a neighborhood for a feasible reference solution and, furthermore, the possibility of generating pseudo-cuts, to exclude parts of the solution space.

The LB strategy requires that a certain number of variables assume the same values as the reference solution, without setting the other variables. This is accomplished by adding a linear constraint to the original problem. The added constraint, given by:

$$\Delta(x,\overline{x}) \leqslant k \tag{11}$$

defines, according to Fischetti and Lodi (2003), a k-optimal neighborhood N^k of $\overline{x}$.

Let $\overline{S} = \{j : \overline{x_j} = 1\}$ be a set. Therefore:

$$\Delta(x,\overline{x}) = \sum\nolimits_{j\in\overline{S}}(1-x_j) + \sum\nolimits_{j\setminus\overline{S}} x_j \tag{12}$$

The neighborhood N^k, for some integer k, is then defined as:

$$N^k(\overline{x}) = \{x \text{ obtained from } (IP) \quad : \quad \Delta(x,\overline{x}) \leqslant k\} \tag{13}$$

As discussed in Fischetti and Lodi (2003), Expression (12) may be used with a branching rule for (IP), after setting $\overline{S}$. Given $\overline{x}$, the solution space associated with the current node in some enumeration scheme can be partitioned by the following rules:

$$\Delta(x,\overline{x}) \leqslant k \quad (\textit{left branch}) \qquad \textbf{or} \qquad \Delta(x,\overline{x}) \geqslant k+1 \quad (\textit{right branch}) \tag{14}$$

The parameter k should be chosen appropriately to ensure an easier subproblem to solve after adding the left-branch to (IP). Therefore, the neighborhood $N^k(\overline{x})$ should be small enough to be optimized in an acceptable time, as well as large enough in an attempt to find solutions better than $\overline{x}$. This LB approach described so far corresponds to an exact method. The conversion of LB to a heuristic can be achieved by adding the `node_time_limit` parameter, which is used to incorporate a timeout for the operation of the left-branch node. This parameter sets the timeout for solving each subproblem generated by the left branch constraint.

Fig. 1 shows a representation of the LB strategy. The search process is described hereafter. Initially, there is a reference solution $\overline{x_1}$ for node "1". Then, the linear pseudo-cut $\Delta(x,\overline{x_1}) \leqslant k$ must be added, thus generating subproblem "2", which is solved at optimality (highlighted by the symbol "*"), finding the new reference solution $\overline{x_2}$. Therefore, the pseudo-cut $\Delta(x,\overline{x_1}) \leqslant k$ is removed in order to add permanently the inequality $\Delta(x,\overline{x_1}) \geqslant k+1$ and, as a consequence, there is the new subproblem "3". By adding the pseudo-cut $\Delta(x,\overline{x_2}) \leqslant k$, the subproblem "4" is solved, finding a feasible and improvement solution $\overline{x_3}$. Since this last subproblem was not solved at optimality (highlighted by the "+" symbol), the previous pseudo-cut is removed to add the pseudo-cut $\Delta(x,\overline{x_3}) \leqslant k$, which generates the subproblem "5", solved at optimality, thereby obtaining the new reference solution $\overline{x_4}$. Thus, the previous pseudo-cut is removed for the addition of a further permanent pseudo-cut $\Delta(x,\overline{x_3}) \geqslant k+1$. It is important to note that, up to this moment, two pseudo-cuts have been added to the *(IP)* model. In subproblem "7", the solver terminated its execution due to the

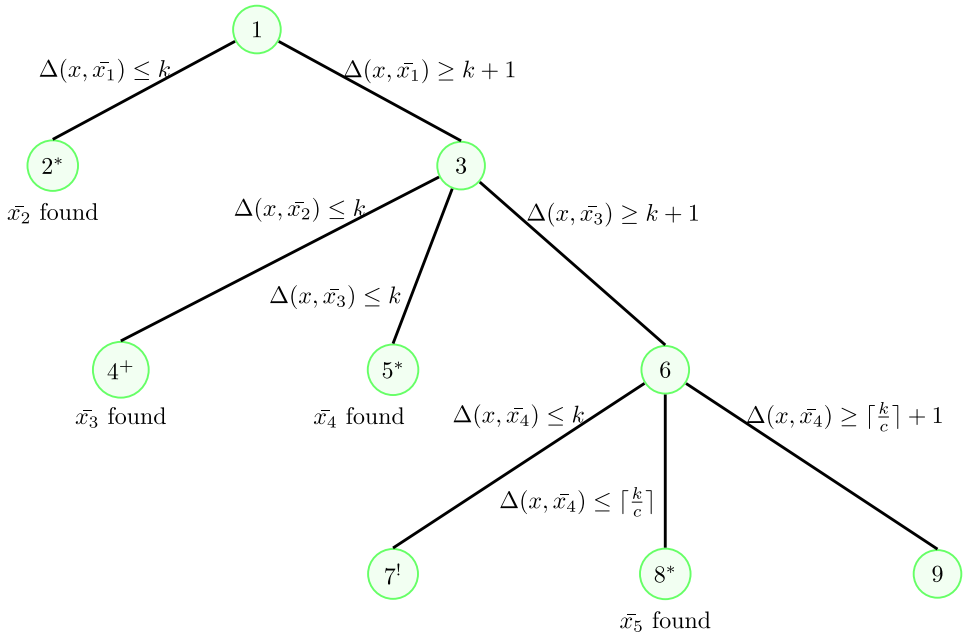


Fig. 1. LB strategy enumeration scheme.

`node_time_limit` parameter, without finding an improvement solution (highlighted by "!"). At this moment, a neighborhood decrease occurs, controlled by the parameter k and a positive constant c, which causes the addition of the pseudo-cut $\Delta(x,\overline{x_4}) \leqslant \left\lceil \frac{k}{c} \right\rceil$, generating the subproblem "8" that is smaller than the subproblem "7". The subproblem "8" is solved at optimality, which causes the permanent addition of the pseudo-cut $\Delta(x, \overline{x_4}) \geqslant \left\lceil \frac{k}{c} \right\rceil + 1$, and, thus, the LB strategy continues to explore the solution space.

4.3. Proposed algorithm

This subsection presents the core of this research, a proposed algorithm for the enumeration scheme illustrated above. The algorithm, based on the LB matheuristic, has the following general operating procedure:

(i) Find an initial and feasible solution for MRCPSP. This search is based on two steps:
  (a) *(IP)* resolution to find feasible execution modes for activities. Tasks are scheduling by initial assignment of execution modes;
  (b) Local search in the initial solution built. This local search also considers the precedence constraint and the time window for activity scheduling.
(ii) Execution of the Local Branching strategy on the feasible initial solution.

The following subsections describe these steps in detail.

4.3.1. Building an initial solution

A feasible assignment of execution modes to the activities to MRCPSP consists of selecting execution modes, such that the consumption of nonrenewable resources is respected, according to the availability of these resources. Using (P) formulation, given by Expressions (1)–(6), it is possible to obtain another formulation, which considers only the assignment of execution modes to the activities. For this, the finish times can be disregarded. Let $x_{jm} = 1$, if the activity j is executed in mode m; and $x_{jm} = 0$, otherwise. The binary formulation (P') is used to obtain a feasible assignment of execution modes to the activities:

$$(P') \quad \min \quad \sum_{j=1}^{n} d_{jm} x_{jm} \tag{15}$$

s.t.

$$\sum_{m=1}^{|M_j|} x_{jm} = 1, j = 1, \ldots, n \tag{16}$$

$$\sum_{j=1}^{n} \sum_{m=1}^{|M_j|} r^{\nu}_{jmk} x_{jm} \leqslant R^{\nu}_{k}, \quad k = 1, \ldots, R^{\nu} \tag{17}$$

$$x_{jm} \in \{0, 1\}, \quad j = 1, \ldots, n; \; m = 1, \ldots, |M_j| \tag{18}$$

According to this formulation, Expression (15) looks for the execution modes that have the shortest duration. Expression (16) guarantees an execution mode by activity, and Expression (17) ensures that the execution modes respect the availability of nonrenewable resources. Expressions (16) and (17) are also shown in (P) formulation of MRCPSP, but, in the current case, do not consider finish times for activities.

Finding an optimal solution for (P') is an NP-Hard problem because this problem corresponds to the well known Multidimensional 0-1 Knapsack Problem (Fréville, 2004). However, current integer programming solvers can solve this problem at optimality at a surprisingly fast computational time, as discussed by Santos, Soares, and Tóffolo (2014).

After setting the execution modes, it is enough to scheduling the activities, respecting the constraints of precedence and consumption of renewable resources. The Serial Scheduling Scheme (SSS) method, described in Kolisch (1996), is used to define the start (and finish) times of activities. SSS method needs a list of activities L_J to create the schedule, respecting the order in which activities appear in the list since this scheduling scheme assigns the earliest possible instant to a given activity. Hence, with the same list of activities, but considering the activities in a different order and respecting the precedence constraints, it may construct more than one distinct schedule (Demeulemeester, 2002).

Let $\overline{x}$ be the solution constructed by SSS, after finding the feasible execution modes while solving (P'). The coding $L(\overline{x}) = (L_J, L_M, L_T)$ represents the solution $\overline{x}$ in lists, that is, L_J for the activities, L_M for the execution modes, and L_T for the finish times. Thus, this coding in lists indicates that, during the SSS processing for a activity j at the position i

of L_J, its execution mode at the position i of L_M is checked to assign a finish time at the i position of L_T.

4.3.2. Local search

To perform a local search process on the solution $\overline{x}$ obtained after the resolution of $(P^{'})$, the coding $L(\overline{x})$ is used by means of neighborhood structures, which were adapted from Asta, Karapetyan, Kheiri, Özcan, and Parkes (2016). Some of these neighborhood structures are based on the First Improvement Local Search (FILS) heuristic (Hoos & Stützle, 2004). These neighborhood structures are:

(i) NJ-SW: this neighborhood is based on the FILS heuristic. An activity j is randomly selected, then, its nearest predecessor p, such that $(p,j) \in \mathcal{P}$ and its nearest successor s, such that $(j,s) \in \mathcal{S}$ are found, respectively, in the positions i_p and i_s of the list L_J. Hence, an interval $I = [i_p + 1, i_s - 1]$ is determined to swap activity j with each activity $\bar{j} \in I$, such that $j \neq \bar{j}$. In other words, the activities j and $\bar{j}$ are swapping positions, if and only if there is no violation of the precedence constraint. If the new value of makespan decreases, then it is not necessary to check the other remaining activities in I, and the process continues until all activities $j = 1, \ldots, n$ are processed. In this work, $|I| = n$, where n is the number of actual project activities;

(ii) NM-CMF: this neighborhood is also based on the FILS heuristic. An activity j is randomly selected. Then all possible execution modes of j are checked for assignment. For an execution mode to be accepted, it must be feasible concerning the consumption of nonrenewable resources. After verifying one execution mode, if it provides a better makespan for the new solution, then it is not necessary to check the other execution modes of j, and the process continues until all activities $j = 1, \ldots, n$ be processed;

(iii) NJ-SH: an activity j is randomly selected. Then the values p and s are determined to identify how far j may move forward or backward in the L_J list. Thus, j is placed in a new position $\bar{j}$, randomly selected from the L_J list, such that $\bar{j} \neq i$ and $i_p < \bar{j} < i_s$;

(iv) NM-CM: an activity j is randomly selected. Then a new execution mode $\overline{m} \neq m$ is checked, where m is the current execution mode for j. To m be changed for $\overline{m}$, it needs to be feasible with respect to the nonrenewable resources consumption.

We choose the neighborhood names by associating them with your own action. For example, the first neighborhood was named NJ-SW, as it is a neighborhood that applies swap move between jobs. The name NM-CMF means a neighborhood applied only to modes, and the move is to change modes while using the FILS heuristic. The name NJ-SH is a neighborhood applied only to jobs, and the move is a shift of an activity position. Finally, NM-CM is a neighborhood applied only to modes, and the move is to change modes associated with a given job.

In the process of constructing a feasible initial solution, the $(P^{'})$ problem is solved, and, thus, the initial feasible solution $\overline{x}$ is obtained, which will be coded in $L(\overline{x})$. In the following, the local search on $\overline{x}$ is performed in two stages. First, a local search on the activities is performed, i.e., the FILS-based NJ-SW neighborhood structure is used. Then, in the second stage, a local search is performed in the execution modes employing the neighborhood structure NM-CMF, which is also based on FILS. The neighborhood structures NJ-SH and NM-CM will be addressed ahead (see description of Algorithm 1).

4.3.3. Local branching matheuristic for MRCPSP

The matheuristic procedure used in this work is an adaptation of the method described above, using $\overline{x}$ as the initial reference solution, obtained after the resolution of $(P^{'})$, and, possibly, improved by the local search procedure.

Fischetti and Lodi (2003) define the support set $\overline{S}$ as $\overline{S} = \{j : \overline{x_j} = 1\}$. Suppose that $\overline{x}$ is the reference solution, such that, whenever $\overline{x}_{jmt} = 1$, then an activity j is performed in an execution mode m and is finished at time t. In the current paper, we characterize the definition of this support set in two distinct ways in the context of MRCPSP. These characterizations are shown below. Hereafter, we discuss another characterization of the support set proposed by Zhu et al. (2006).

4.3.4. Proposed subproblem characterization

In the context of MRCPSP, we can define $\overline{S}$ as:

$$\overline{S} = \{t : EFT_j \leqslant t \leqslant LFT_j, \quad \overline{x}_{jmt} = 1\} \tag{19}$$

From Expression (19), the neighborhood of a reference solution $\overline{x}$ is characterized by solutions that differ from $\overline{x}$ for different activities finish times t. This difference is allowed up to a k degree, according to $N^k(\overline{x})$ as defined in Expression (13). This characterization for $\overline{S}$ is restricted, since, in the definition of $N^k(\overline{x})$ from Expression (13) by left branch, no solution with execution mode other than the reference solution is allowed, which reduces MRCPSP to RCPSP when $N^k(\overline{x})$ is solved.

On the other hand, concerning the second interpretation, $\overline{S}$ is given by:

$$\overline{S} = \{(m,t) : EFT_j \leqslant t \leqslant LFT_j, \quad \overline{x}_{jmt} = 1\} \tag{20}$$

In this case, the neighborhood of a reference solution $\overline{x}$ is characterized by solutions that differ from $\overline{x}$ concerning different execution modes m and activities finish times t. It makes this latter characterization, from Expression (20), more extensive than the first, given by Expression (19). Each subproblem generated by the characterization from Expression (20) becomes larger than the previous characterization from Expression (19). The set $\overline{S}$ for MRCPSP was defined similarly by Zhu et al. (2006).

4.3.5. Subproblem characterization by Zhu et al. (2006)

A B&C-based approach to the MRCPSP solution is presented by Zhu et al. (2006). These authors define the support set $\overline{S}$ as:

$$\overline{S} = \{t : \max(EFT_j, t-\delta) \leqslant t \leqslant \min(LFT_j, t+\delta), \quad \overline{x}_{jmt} = 1\}, \tag{21}$$

where m is the execution mode of activity j in the reference solution; t is the finish time of activity j in the reference solution, for some $\delta \geqslant 0$. That is, the neighborhood of a reference solution $\overline{x}$ is characterized by solutions that differ from $\overline{x}$ concerning different activities finish times t, but within the time-window $[\max(EFT_j, t-\delta), \min(LFT_j, t+\delta)]$. The set $\overline{S}$ of Expression (21) includes variables that are at a δ distance from activities completion times in the reference solution. Note that this definition of Zhu et al. (2006) also reduces MRCPSP to RCPSP.

Since the purpose of the current article is to investigate the influence of the characterization of an inequality class, Expression (21) is also explored to define $\overline{S}$ as:

$$\overline{S} = \{(m,t) : \max(EFT_j, t-\delta) \leqslant t \leqslant \min(LFT_j, t+\delta), \quad \overline{x}_{jmt} = 1\}, \tag{22}$$

that is, the neighborhood of a reference solution $\overline{x}$ is characterized by solutions that differ from $\overline{x}$ in relation to different execution modes m and activity finish times t, but into the temporal window $[\max(EFT_j, t-\delta), \min(LFT_j, t+\delta)]$. This description expands the definition from Expression (21).

4.3.6. Local branching algorithm

Let $\overline{x}$ be the built initial feasible solution and X the set of all feasible solutions of (P). Thus, the search in a neighborhood $N^k(\overline{x})$ can be defined by adding the pseudo-cut $k_{min} \leqslant \Delta(x,\overline{x}) \leqslant k$. The new set of solutions $X = X \backslash N^k(\overline{x})$ is represented by eliminating $k_{min} \leqslant \Delta(x,\overline{x}) \leqslant k$, and by adding the pseudo-cut $\Delta(x,\overline{x}) \geqslant k + 1$. Algorithm 1 shows the Local Branching matheuristic adapted for solving MRCPSP.

Algorithm 1
Adapted Local Branching - (LB)

Data: X, $\bar{x}$, k_0, β, node_time_limit, stop_criterion
Result: $\bar{x}$
1 $k \leftarrow k_0$;
2 $k_{min} \leftarrow 1$;
3 **while** *stop_criterion is not satisfied* **do**
4 $x^* \leftarrow$ MIP_SOLVER$(P, X \cap N^k(\bar{x}),$ node_time_limit) ;
5 **if** *($x^* \in \{X \cap N^k(\bar{x})\}$ is optimal or feasible) AND (x^* is better than $\bar{x}$)* **then**
6 **if** *x^* is optimal* **then**
7 $X \leftarrow X \backslash N^k(\bar{x})$;
8 *reset* k and $k_{\min}$;
9 **end**
10 $\bar{x} \leftarrow x^*$;
11 **else**
12 **if** *(no feasible solution $x^* \in \{X \cap N^k(\bar{x})\}$ is found) OR (x^* is optimal and it is not better than $\bar{x}$)* **then**
13 $k_{min} \leftarrow k$;
14 $k \leftarrow k + \max(1, \lceil k/2 \rceil)$;
15 **else**
16 $k \leftarrow \max(1, \lceil k/2 \rceil)$;
17 **if** $k \leq k_{\min}$ **then**
18 $\bar{x} \leftarrow$ StrongDiversification$(\bar{x}, \beta, n)$;
19 **end**
20 **end**
21 **end**
22 **end**
23 **return** $\bar{x}$

Algorithm 1 begins with a feasible solution $\bar{x}$ of (P). Then the node corresponding to the left branch is added, creating a subproblem solved by a MIP solver. Three cases are possible after the resolution of the MIP subproblem:

(i) Case 1: if an optimal and improvement solution x^* is found, then the process performs a backtrack to the previous node, and the left branch is replaced by the right branch in (P). From that instant, the reference solution is updated to x^*, and a new left branch is added. If x^* is an improvement solution, but not optimal, then there is only the update of the reference solution. This first condition is shown between lines 5 to 10 of Algorithm 1;
(ii) Case 2: if the solution x^* is optimal without showing improvement in relation to $\bar{x}$, or if no feasible solution is found, then the neighborhood $N^k(\bar{x})$ becomes larger due to the increase of k and, consequently, the LB method is expected to perform a new search in an increased neighborhood, without the addition of a right branch, because there are no mathematical criteria that allow it to exclude the region created by the left branch. However, a new left branch will be created for an increased k. This condition is shown between lines 12 to 14 of Algorithm 1;
(iii) Case 3: if the timeout is reached and x^* is infeasible, or it is not a better solution than $\bar{x}$, the neighborhood cardinality is decreased in an attempt to guarantee local optimality in a smaller neighborhood generated by the new left branch for a reduced k. This event is shown between lines 16 to 19 of Algorithm 1.

When the neighborhood cardinality is decreased using k, the LB matheuristic could enter in a loop. This event occurs for values of k equal to or less than $k_{\min}$. If this happens, a perturbation procedure is required for the diversification of the LB method. This procedure is highlighted by Fischetti and Lodi (2003) as *Strong Diversification*. This perturbation procedure is necessary to remove the LB strategy from a local optimum, generating a new reference solution. Algorithm 2 shows this process of *Strong Diversification*.

If Case 3 holds, the neighborhood cardinality is decreased, and the LB matheuristic could enter in a loop. This event occurs for values of k equal to or less than $k_{\min}$. Consequently, a perturbation procedure is required for the diversification of the LB method. This procedure is highlighted by Fischetti and Lodi (2003) as *Strong Diversification*. This perturbation procedure is necessary to remove the LB strategy from a local optimum, generating a new reference solution. Algorithm 2 shows this process of *Strong Diversification*.

The diversification procedure used in this work includes the neighborhood structures: NJ-SH, NJ-SW, NM-CM, and NM-CMF. The neighborhood structures NJ-SH and NM-CM will modify the solution $\bar{x}$. This alter the current solution $\bar{x}$ by at most β%, according to line 1, hence performing modifications in the order of activities and modifications in the execution modes, obtaining then the solution $\bar{x}_1$ modified. These modifications will be at most the number of activities n in the project. For $\bar{x}$ to be modified, first, $L(\bar{x})$ encoding is generated from $\bar{x}$, as shown by line 2; in the following, the neighborhoods NJ-SH and NM-CM alter the solution $L(\bar{x})$ as exposed between line 3 and line 6; and the temporary list-coded solution $L(\bar{x}_1)$ is produced. After the perturbation, the

Algorithm 2
StrongDiversification

Data: $\bar{x}, \beta, n$
Result: $\bar{x}_2$
1 $n\_mod \leftarrow \lceil \beta/100 \rceil * n$;
2 encode $\bar{x}$ to $L(\bar{x})$;
3 **for** $i \leftarrow 1$ *to* n_mod **do**
4 modifies $L(\bar{x})$ by NJ-SH ;
5 modifies $L(\bar{x})$ by NM-CM ;
6 **end**
7 $L(\bar{x}_1) \leftarrow FILS(\text{NJ-SW}, L(\bar{x}))$;
8 $L(\bar{x}_2) \leftarrow FILS(\text{NM-CMF}, L(\bar{x}_1))$;
9 decode $L(\bar{x}_2)$ to $\bar{x}_2$;
10 **return** $\bar{x}_2$;

Table 2
PSPLIB classes.

Class	J10	J12	J14	J16	J18	J20	J30
# feasible instances	536	547	551	550	552	554	552
# executions modes	3	3	3	3	3	3	3
# renewable resources	2	2	2	2	2	2	2
# nonrenewable resources	2	2	2	2	2	2	2

Table 3
MMLIB classes.

Class	J50	J100
# feasible instances	540	540
# executions modes	3	3
# renewable resources	2	2
# nonrenewable resources	2	2

$L(\bar{x}_1)$ list-coded solution passes through the local search process, using the neighborhoods NJ-SW and NM-CMF, based on the FILS heuristic, as shown by both lines 7 and 8, generating the new list-coded solution $L(\bar{x}_2)$. The new solution $\bar{x}_2$ is finally found decoding $L(\bar{x}_2)$, as shown in line 9.

Note that both the neighborhood structures that perform the perturbation and the neighborhood structures that carry out the local search are simple and fast structures in their executions, which provides to the LB method a greater time-period released to explore the solution space in an attempt to guarantee local optimality. This is an important feature of this current proposal.

5. Computational results

The adapted Local Branching matheuristic proposed in the current paper is implemented in C++ language, using the CPLEX 12.6 solver and GNU G++ compiler 5.4.0. Two computational tests are realized. The first one comprises the public library PSPLIB and is executed in an Intel Core i3-3217U processor; 1.80 GHz; 4 GB RAM; and Ubuntu Linux 64-bit operating system. The second computational test uses the public library MMLIB and is executed in an Intel Core i7-3770U processor; 3.40 GHz; 32 GB RAM; and Ubuntu Linux 64-bit operating system. Tables 2 and 3 present the characteristics of these libraries.

This section starts in Section 5.1, by introducing the parameter tuning procedure performed; then, in Section 5.3, the obtained computational results are presented, as well as comparisons are carrying out with the literature results concerning the MRCPSP solution for the addressed instances.

Table 4
Set of the first combination of the parameters level.

Parameters	Levels
node_time_limit	(**120**, 180)
k_0	(10, 15, 20, **25**, 30)
β	(20, 30, 40, **50**)

Table 5
Set of the second combination of the parameters level.

Parameters	Levels
node_time_limit	(60, **90**, 100, 120)
β	(50, **80**, 100)

5.1. Parameters tuning

The parameter tuning has an important influence on the results obtained by computational procedures. In the current article, we used the IRACE method (López-Ibáñez, Dubois-Lacoste, Cáceres, Birattari, & Stützle, 2016) to tune the parameters of the implemented matheuristic. The IRACE method automatizes a manual DOE for configuring the parameters of an optimization algorithm. For this purpose, we adopted the experiment described in the following.

We randomly selected 200 instances of class J20 from the PSPLIB set of instances. As this class is the larger one of PSPLIB with the known optimal values, we have chosen this class in this initial experiment. Each selected instance was solved once for, then, analyzing the average deviation as the response variable for the known optimum value. The stop criterion for the LB adapted matheuristic was set to 1800 s. Among the 200 instances solved, 100 are classified with Resource Factor (RF) equal to 1, and another 100 were classified as RF equal to 0.5. This factor indicates the demanded quantity, on average, of each resource, for each activity-mode combination. If $RF = 0$, each activity-mode combination does not consume resources; on the other hand, if $RF = 1$, each activity-mode combination consumes some units of each type of resource. According to the literature, this factor indicates how difficult it is to solve an instance.

The matheuristic presented in Algorithm 1 has three parameters: (i) node_time_limit; (ii) k_0; and (iii) β. Two sets of values are used for analyzing the influence and tuning these parameters with IRACE. The first set of values used for these parameters is shown in Table 4. These values were chosen empirically, after some previous tests. The values highlighted in bold in Table 4 exhibited the best values for the average deviation from the known optimal solution, according to IRACE tuning. This first round of IRACE allowed us to set only the parameter k_0. The best values provided for the node_time_limit and β parameters were the lower and the upper limit values, respectively. Consequently, we performed the second round of IRACE to tuning these parameters, with the the second set of values, as shown in Table 5. These values were chosen accordingly to our observation of the trend values in the first set. Again, the values provided in bold in Table 5 exhibited the best values for the average deviation from the known optimal solution, according to

Table 6
Influence of Local Search FILS on the Adapted LB Algorithm.

Procedure	#*opt*	%*opt*	%gap_{opt}
LB + FILS CONS + FILS SD	530	96.0	0.1393
LB + FILS CONS	515	93.0	0.3589
LB	507	91.0	0.3699

Table 7
Results for PSPLIB (J10-J30) using Expressions (19 and 20).

	$\overline{S}$ defined by Expression (19)				$\overline{S}$ defined by Expression (20)			
	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$
J10	536	100	0	–	536	100	0	–
J12	547	100	0	–	547	100	0	–
J14	551	100	0	–	551	100	0	–
J16	544	98.9091	0.1112	–	550	100	0	–
J18	528	95.6522	0.1955	–	547	99.0	0.040	–
J20	518	93.5018	0.2839	–	530	96.0433	0.1351	–
J30	438	74.7246	1.8591	15.6529	453	82.5362	1.2383	13.1771

Table 8
Results for PSPLIB (J10-J30) using Expressions (21 and 22).

	$\overline{S}$ defined by Expression (21)				$\overline{S}$ defined by Expression (22)			
	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$	$\#opt$	$\%opt$	$\%gap_{opt}$	$\%gap_{cpm}$
J10	536	100	0	–	536	100	0	–
J12	547	100	0	–	547	100	0	–
J14	551	100	0	–	551	100	0	–
J16	544	98.9391	0.0921	–	545	99.0909	0.0885	–
J18	536	97.1014	0.1027	–	533	96.5580	0.1197	–
J20	521	94.0433	0.2353	–	517	93.3213	0.2902	–
J30	447	80.9783	1.2295	14.0899	433	78.4420	1.5389	14.5256

IRACE tuning. The characterization of the support set $\overline{S}$ for parameter tuning was performed by Expression (20).

In the following, considering that the parameters of the LB adapted matheuristic were set as `node_time_limit`$= 90, \beta = 80$ and $k_0 = 25$, we carried out a computational experiment to evaluate the influence of the local search performed only on the construction phase of an initial solution (LB + FILS CONS); the local search performed on both the construction phase of an initial solution and on the diversification phase (LB + FILS CONS + FILS SD); and without the local search (LB). In this case, all J20 instances from PSPLIB were solved, with a stop criterion of 1800 s. Table 6 displays the obtained result. In this Table, column $\#opt$ shows the number of found optimal solutions; column $\%opt$ shows the percentage of found optimal solutions against the total known optimal solutions of this instance family, and column $\%gap_{opt}$ shows the average deviation of the found values against the known optimal values. As observed, the use of the local search FILS brought advantage to the LB adapted method, contributing to a greater number of found optimal solutions ($\#opt$ and $\%opt$) and a lower average deviation ($\%gap_{opt}$) from the known optimum values.

This difference in favor of the use of the local search shows that, when constructing a new solution $\overline{x}$, the local search improves this solution and, thus, the neighborhood $N^k(\overline{x})$ is more quickly explored to prove local optimality of $\overline{x}$, and then to exclude $N^k(\overline{x})$ from the solution space, thereby decreasing it with each new right branch created.

5.2. Subproblems characterizations

Another computational experiment was performed to compare the influence of the characterizations presented by Expressions (19)–(22).

Table 9
Average deviations obtained for PSPLIB (J20 and J30) by LB variant using Expressions (20) and (21).

		J20		J30	
		$RF = 0.5$	$RF = 1$	$RF = 0.5$	$RF = 1$
Expression (20)	$\%gap$	0.000000	0.277197	0.288364	2.009556
Expression (21)	$\%gap$	0.023391	0.441729	0.336282	2.088222

For this experiment, the instances J10, J12, J14, J16, J18, J20, and J30 from PSPLIB were solved. Table 7 presents the values obtained by the LB procedure using Expressions (19) and (20). Table 8 shows the values obtained using Expressions (21) and (22) with $\delta = 1$. In these tables, in addition to the notations already previously defined, the column $\%gap_{cpm}$ expresses the average deviation of the solutions to the critical path lower bound.

According to Table 7, the LB variant using Expression (20) has a better resolvability of the PSPLIB instance set than the LB variant using Expression (19). This fact becomes clear for J16, J18, J20, and J30 instance classes, in which the LB variant using Expression (20) exhibits a greater number of optimal values and lower *gap* values for the optimal values than the other variant. As the LB variant using Expression (20) has a larger neighborhood structure, the LB matheuristic can evaluate a greater number of solutions during the search process, which can provide better solutions. It is worth mentioning the results of class J30, in which the found number of optimum values is ten percent greater than that found by the other variant, in addition to a 33% lower average gap value. Therefore, the MRCPSP structure is better exploited when we characterized a subproblem through Expression (20).

On the other hand, Table 8 shows that the LB variant using Expression (21) exhibit, on average, for J18, J20, and J30 instance classes, better results when compared to the LB variant characterized by Expression (22). These best results consist of determining a greater number of optimal values and obtaining lower *gap* values concerning the optimal values in the literature. Therefore, for these characterizations, from Zhu et al. (2006) and already described in Section 4.3.5, the one that has the lowest neighborhood cardinality presented the best results. It should be noted, however, that the difference between the values for the two characterizations is smaller than those presented in the analysis shown in Table 7.

A direct conclusion of great importance, according to the results shown in Tables 7 and 8, is that the definition of the neighborhood by Expression (20) used in the LB variant presented better results in the resolution of the PSPLIB instances, even surpassing the characterization proposals of Zhu et al. (2006).

For a better evaluation of the obtained results, we separated the J20 and J30 instance classes by groups, according to the RF demand. Table 9 presents the results concerning the average deviations. Note that if the

Table 10
Runtime trend for PSPLIB (J20-J30) using Expressions (20).

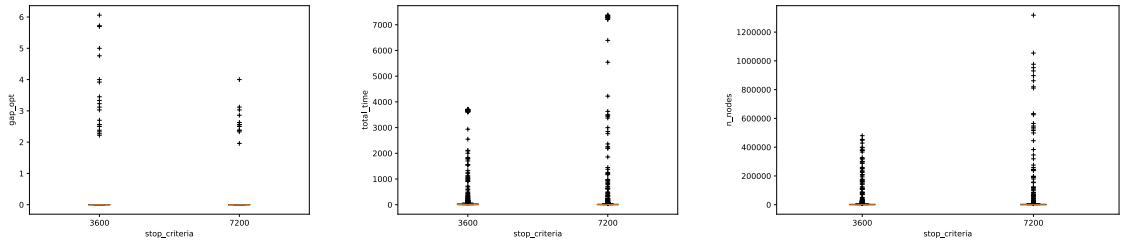
J20							
(ntl, stc)	*opt* (%)	gap_{opt} (%)	gap_{cpm} (%)	tot_rt (sec.)	ls_rt (sec.)	avg_nodes	avg_psc
(90, 1800)	96.0433	0.1351	–	168.846	0.0001	16409	6
(120, 3600)	97.2094	0.1182	–	242.142	0.0001	22652	7
(120, 5400)	98.4534	0.0818	–	252.276	0.0002	25930	7
(180, 7200)	98.9534	0.0630	–	320.34	0.0001	36621	7
J30							
(ntl, stc)	*best* (%)	gap_{best} (%)	gap_{cpm} (%)	tot_rt (sec.)	ls_rt (sec.)	avg_nodes	avg_psc
(90, 1800)	82.5362	1.2383	13.1771	607.392	0.006	37639	15
(120, 3600)	84.0797	1.1924	12.2970	1174.254	0.006	53465	21
(120, 5400)	86.8913	1.0741	11.1401	1669.896	0.012	77251	28
(180, 7200)	89.4275	0.8793	10.7324	1912.572	0.007	95047	23

number of activity-mode pairs requiring resources in an instance is large, it is harder to solve it. According to these results, the neighborhood definition by Expression (20), used in the LB variant and proposed in the current article, was again the one that presented the best results by solving PSPLIB, concerning the proposed characterization by Zhu et al. (2006). Note that, for the two instance classes and the two adopted RF values, the average percentage *gap* values of the neighborhood definition by Expression (20) are lower than the values of the neighborhood definition by Expression (21). This result shows, thus, the superiority of the LB variant using Expression (20).

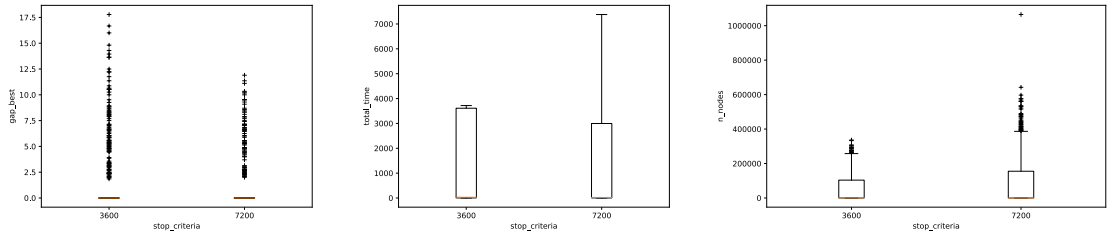
To fully characterize the behavior of the adapted LB matheuristic when using Expression (20), we performed a computational test by varying parameters `stop_criterion` (named, for briefness, as `stc`) and `node_time_limit` (named here as `ntl`) for solving classes J20 and J30, i.e., the largest instances of PSPLIB. We considered that `stc` assumed the values 1800, 3600, 5400, and 7200 s, while `ntl` assumed the values 90, 120, and 180 s. Through these variations, we analyzed the average runtime for each class of problems (`tot_rt`), the average runtime of the *Strong Diversification* procedure (`ls_rt`), the average number of nodes explored by the CPLEX solver (`avg_nodes`), and the average number of pseudo-cuts (`avg_psc`) given by Expression (20). Therefore, with this analysis, it is possible to infer the trend of the adapted LB matheuristic proposed in this article by varying its main parameters. Table 10 summarized the results.

From Table 10, we can see that, as the adapted LB matheuristic has more time to explore each subproblem and more time as a stop criterion, this method finds better solutions, which can be evidenced by the first three columns of this Table. It is noted that in all tests, the runtime of the *Strong Diversification* procedure is very small concerning the total runtime, that is, as previously exposed, this procedure is quite fast in its execution, so that the largest portion of time remains to explore a subproblem after a pseudo-cut is added. Table 6 has already pointed out the importance of this procedure for the success of the adapted LB matheuristic. Also noteworthy is the total runtime of the adapted LB matheuristic, which, on average, for all problems, in class J20 did not exceed 25% of the runtime given for execution, and for class J30 it does not exceed 40% of the time given for the stop criterion. Therefore, the adapted LB method tends to solve the instances by finding the optimal solution earlier than the time given for the execution of this method.

Fig. 2 presents these last values for a better analysis of all these factors. By this figure, it is clear that `tot_rt` and `avg_nodes` are the factors that show the greatest growth as we increase the execution time of the adapted LB matheuristic while solving J30. On the other hand, % gap_{opt} and %gap_{best} do not show much variation by increasing the



(a) J20: gap_{opt} (b) J20: Total runtime (c) J20: Number of nodes



(d) J30: gap_{best} (e) J30: Total runtime (f) J30: Number of nodes

Fig. 2. Runtime trend concerning solving PSPLIB J20 and J30 instances, using adapted LB matheuristic with Expressions (20), and stopping criteria of 3600 and 7200 s.

Table 11
Comparison between the results for J20 and J30 PSPLIB classes instances achieved in Zhu et al. (2006), G12Max and SCIPMax from Schnell and Hartl (2017), and the adapted LB matheuristic using Expression (20). Symbol "–" refers to unpublished results.

Article	J20 (554 instances)		J30 (552 instances)		
	$\#opt$	$\%gap_{opt}$	$\#best$	$\%gap_{best}$	$\%gap_{cpm}$
Zhu et al. (2006)	554	0	529	–	–
G12Max[a]	552[b]	0.05[b]	521[c]	2.59[c]	12.93[c]
SCIPMax[a]	547[b]	0.32[b]	508[c]	3.37[c]	13.32[c]
This work[d]	532	0.1351	455	1.2383	13.1771
This work[e]	548	0.0630	494	0.8793	10.7324

[a] From Schnell and Hartl (2017).
[b] Stopping criterion of 1200 s.
[c] Stopping criterion of 2400 s.
[d] Results shown in Table 10 for $(ntl, stc)=(90, 1800)$.
[e] Results shown in Table 10 for $(ntl, stc)=(180, 7200)$.

execution time. Finally, for class J20, there was no significant variation when analyzing all factors so that the LB method was stable to solve this class of problems.

5.3. Results in relation to literature

In this section, we show the comparison between the results achieved by the adapted matheuristic LB with the characterization of the support set by Expression (20), proposed in the current article, and the state-of-the-art results from the literature, concerning to solve instances of the PSPLIB and MMLIB public libraries. Section 5.3.1 shows the results regarding PSPLIB; Section 5.3.2 presents the results associated with MMLIB.

It is worth mentioning that, in the literature referring to MRCPSP, the articles that use (meta)heuristic methods typically use, as the stop criterion, the number of generated scheduling, as discussed in Geiger (2017) and Schnell and Hartl (2017). On the other hand, the articles that use models based on mathematical programming or heuristics hybridized with mathematical programming generally use runtime as the stop criterion (Schnell & Hartl, 2017).

Table 12
Comparative results concerning the state-of-the-art metaheuristics for J20 and J30 PSPLIB classes instances. Results from literature refer to 5000 and 50000 generated schedules. The symbol "–" refers to unpublished results.

Article	J20 (554 instances)			
	$\%opt$		$\%gap_{opt}$	
	5000	50000	5000	50000
Van Peteghem and Vanhoucke (2014)	87.73	–	0.32	–
Muritiba et al. (2018)	97.83	99.64	0.06	0.01
Chakrabortty et al. (2020)	–	–	0.27	–
This work[a]	97.2094[b]	98.9534[c]	0.1351[b]	0.0630[c]

Article	**J30** (552 instances)			
	$\%best$		$\%gap_{cpm}$	
	5000	**50000**	**5000**	**50000**
Van Peteghem and Vanhoucke (2014)	–	–	13.66	12.72
Muritiba et al. (2018)	–	-	12.85	12.55
Chakrabortty et al. (2020)	–	-	13.63	–
This work[a]	84.0797[b]	89.4275[c]	13.1771[b]	10.7324[c]

[a] The results of the current article are the ones shown in Table 10.
[b] `ntl, stc`=(120, 3600).
[c] `ntl, stc`=(180, 7200).

5.3.1. Results for the PSPLIB instance classes

For the discussion of the results obtained by the current article for PSPLIB instance classes, the computational results of Zhu et al. (2006) and Schnell and Hartl (2017) were defined as references. These articles are considered as state-of-the-art for applying mathematical programming in the MRCPSP solution. Both Zhu et al. (2006) and Schnell and Hartl (2017) define the runtime of their approaches as the stop criterion. Schnell and Hartl (2017) describe six approaches based on Constraint Programming (CP), with some of these approaches being executed in parallel processing. The best result of these authors, without the use of parallel processing, was defined as a reference. Both Zhu et al. (2006) and Schnell and Hartl (2017) used the J20 and J30 classes from PSPLIB with the same value of runtime as the stop criterion. Regarding the computational architecture, Zhu et al. (2006) used a Intel Xeon @1.80 GHz processor, that is, having the same processing frequency as the processor we used to perform the computational tests for the PSPLIB instances. Schnell and Hartl (2017) used a two six-core Intel Westmere X5650 @2.66 GHz processors with 24 GB RAM.

Table 11 displays the results presented in these two articles in comparison to those obtained by the procedure proposed here. In this table, $\#opt$ refers to the number of optimal solutions found; $\#best$ refers to the number of best-known solutions found; gap_{opt} refers to the average deviation of solutions for optimal; gap_{best} refers to the average deviations of the solutions for the best-known solution; and, finally, gap_{cpm} refers to the average deviations of the solutions to the critical path lower bound.

There are two main differences between the Branch & Cut algorithm shown in Zhu et al. (2006) and the adapted LB matheuristic presented in the current paper. Zhu et al. (2006), besides proposing an algorithm B&C based on the LB method using the Expression (21) to define the support set, also presented a study of the influence of some valid inequalities on MRCPSP, using these cuts for leading to better limits in the solution of the problem (P). Additionally, Zhu et al. (2006) incorporated a genetic algorithm into the LB matheuristic in the form of a black-box, once the article did not include any description concerning the details of the implemented metaheuristic. This metaheuristic fulfills, in Zhu et al. (2006), important functions of diversification and generation of feasible solutions when necessary. On the other hand, the adapted LB matheuristic proposed in the current paper does not use valid inequalities, but only the pseudo-cuts discussed in subSections 4.3.4 and 4.3.5. Furthermore, the diversification procedure used in this paper is a simple local search based on the FILS heuristic, fully detailed in Algorithm 2. This heuristic requires less computational resources than the genetic algorithm presented in Zhu et al. (2006). It is important to highlight we present other variations of the pseudo-cut used in Zhu et al. (2006), represented by Expression (21). We found that one of the variations proposed in the current paper presents better results than the pseudo-cut presented by Zhu et al. (2006). However, we realized that the use of valid inequalities to solve (P), used by Zhu et al. (2006), provides better solutions during the search process.

As Table 11 shows, the Constraint Programming model displayed in Schnell and Hartl (2017) also provides a greater number of optimal solutions than the LB adapted matheuristic. However, it should be noted that the LB adapted matheuristic, proposed in the current article, concerning the $\%gap_{best}$ value of the J30 class, presented better value than the average deviation found by Schnell and Hartl (2017). This result strengthens the evaluation that the matheuristic proposed in the current article has good performance, evidenced by approximately 1% difference regarding the gaps for the best or the best-known solution.

Table 12 compares the results of the proposal presented here for classes J20 and J30 of the PSPLIB library against the state-of-the-art results in the literature for approaches based on metaheuristics. These approaches are dependent on the number of generated schedules, as already highlighted, this being, even, the stopping criterion used in these cases. For the J20 class, our results prove to be strongly competitive against the best in the literature. For the stopping criterion of 3600

Table 13
Comparison between the results based on the state-of-the-art metaheuristics and the adapted LB matheuristic using Expressions (20) for classes J50 and J100 from MMLIB. Results from literature refer to 5000 and 50000 generated schedules. The symbol "–" refers to unpublished results.

	J50 (540 instances)			J100 (540 instances)		
	#*opt*	%gap_{cpm}		#*opt*	%gap_{cpm}	
Article		5000	50000		5000	50000
Van Peteghem and Vanhoucke (2014)	212[b]	25.45	23.79	236[b]	26.51	24.02
Geiger (2017)	216[b]	33.02	28.35	236[b]	44.11	32.91
Muritiba et al. (2018)	–	24.24	23.52	–	25.18	24.01
Chakrabortty et al. (2020)	220[b]	35.6	31.92	236[b]	65.7	60.24
This work[a]	211	25.6073		134	32.4081	

[a] Stopping criterion equal to 3600 s.
[b] Number of optimal values *apud* Chakrabortty et al. (2020). This number concerning Muritiba et al. (2018) is unpublished.

s, the proposed LB matheuristic found 539 optimal values; for 7200 s, it found 548 optimal values. These results are compatible with those found by Muritiba et al. (2018), the state-of-the-art result in the literature using metaheuristics for this instance class. The percentage gap values for the optimal critical path, likewise, are strongly compatible with those found in the literature.

For class J30, the results found in the literature using approaches based on metaheuristics refer to %gap_{cpm}, i.e., the average deviation of the solutions obtained to the lower limit of the critical path. Thus, we highlight that, for the stopping criterion of 3600 s, the proposal presented here determines 464 optimal values, a value that rises to 494 when the stopping criterion is changed to 7200 s. It is important to remember that there is no optimality guarantee for the results using metaheuristics. Concerning the gap for the critical path, the results of the proposed approach surpassed the state-of-the-art for solutions via metaheuristics, as can be seen when comparing our results against those in Muritiba et al. (2018). However, it is important to note that, on the one hand, there are important differences in the used computational architectures; on the other hand, the proposed approach runtimes are significantly longer when compared to the approaches using metaheuristics.

5.3.2. Results for the MMLIB instance classes

Regarding MMLIB instances, we conducted a comparison to the results from Van Peteghem and Vanhoucke (2014), Geiger (2017), Muritiba et al. (2018), Chakrabortty et al. (2020). We considered that these articles represent the most significant results for the solution, using metaheuristics, of these classes of instances. The stopping criterion of the LB adapted matheuristic was set to 3600 s. The values of the parameters k_0, β, and `node_time_limit` were fixed to $k_0 = 20, \beta = 80$, and `node_time_limit` = 120. Table 13 shows this comparison. In this table, the #*opt* value indicates how many optimal values were obtained equal to the lower limit of the critical path, and the %gap_{cpm} value shows the average deviation of the solutions obtained to the lower limit of the critical path. The objective is to show the potential of the LB adapted matheuristic for solving large and difficult instance classes. All these works from literature adopt the number of generated schedules as a stopping criterion. Van Peteghem and Vanhoucke (2014), Geiger (2017), and Muritiba et al. (2018) show the meaning and influence of this parameter on the quality of the achieved results with their metaheuristics, i.e., the higher the number of generated schedules, the better is the generated solution. These authors, however, point out that this criterion (i) cannot be applied to all different heuristic strategies, and (ii) the required time to compute a schedule might differ between metaheuristics.

Concerning the number of optimal solutions, for J50 class, the adapted LB method found approximately the same number as the metaheuristic procedures are shown; for class J100, we found 24.8% of the optimal solutions, while the metaheuristic procedures determined 43.7% of the optimal solutions for this class. Beyond the differences relating to the stopping criterion, the results from metaheuristic procedures do not directly guarantee the optimum status. Regarding the average deviation of the solutions for the critical path lower bound, for J50 class, the adapted matheuristic we propose surpass the best results provided by Geiger (2017) and Chakrabortty et al. (2020), and, on the other hand, it is only 7.6% and 8.9% worse than the ones from the best Van Van Peteghem and Vanhoucke (2014) and Muritiba et al. (2018) results, respectively, all these for 50000 generated schedules. For the J100 class, also for this number of generated schedules, this result is qualitatively repeated for the same articles, but it is important to highlight the difference for the result presented in Chakrabortty et al. (2020), of the order of 46%. Therefore, the results from the adapted LB method showed competitive with the state-of-the-art results in the literature; however, runtimes are much longer than the ones for metaheuristic procedures.

Table 14 compares the results obtained by the LB adapted matheuristic to the results from two constraint programming models (SCIPMax and G12Max) shown in Schnell and Hartl (2017). These approaches proposed in Schnell and Hartl (2017) guarantee the optimality of its results for several instances of classes J50 and J100. As shown in Table 14, for the J50 class, both have an average deviation value %gap_{cpm} that does not surpass that achieved by the LB adapted matheuristic; additionally, for the J100 class, the SCIPMax approach is again overcome in the average deviation value by the LB adapted matheuristic. Regarding the number of instances solved to optimality represented by #*opt*, SCIPMax constrained programming model outperforms both G12Max and LB adapted matheuristic for the two MMLIB classes. However, it is important to point out the difference of runtime of the methods. Using a factor to adjust the runtime in processing, we conclude that we could have a runtime of 4600 s as the stopping criterion to the adapted LB method proposed in this article. Hence, as discussed previously, the LB method would tend to improve the quality of the solutions found.

Table 14
Comparison between the results of Schnell and Hartl (2017) and the ones of the adapted LB matheuristic using Expressions (20) for classes J50 and J100 from MMLIB.

	J50 (540 instances)			J100 (540 instances)		
	#*opt*	%gap_{cpm}	`tot_rt` (sec.)	#*opt*	%gap_{cpm}	`tot_rt` (sec.)
SCIPMax[b]	**405**	34.21	**1409.69**	**312**	154.19	3127.93
G12Max[b]	363	27.61	1952.36	150	**25.71**	5963.62
This work[a]	211	**25.6073**	2205.11	134	32.4081	**2876.80**

[a] Stopping criterion equal to 3600 s.
[b] Stopping criterion equal to 5400 s, for J50 class, and equal to 7200 s, for J100 class.

6. Conclusion

This article presented an adaptation of the Local Branching (LB) strategy for solving the Multi-mode Resource-Constrained Project Scheduling Problem (MRCPSP). The adaptations of the constraints generated by the standard LB method for the problem under analysis were discussed, as well as an implementation of an LB-based matheuristic to solve MRCPSP was described and evaluated. The PSPLIB and MMLIB public libraries classes were solved to measure and validate the quality of the proposed matheuristic strategy.

Initially, the influence of the parameters of a local search based on the First Improvement Local Search method was verified, as well as the characterization of neighborhood structures that define a subproblem

during the enumeration of the LB matheuristic. We proposed two characterizations to define the subproblems, and also we described two other characterizations from literature. Then, through the results, we verified that one of the characterizations proposed in this article surpasses one of the characterization discussed in the literature. This is one of the main results of this work.

In the following, the results for the PSPLIB classes were compared with the two state-of-the-art literature for the solution of MRCPSP using mathematical programming techniques, i.e., the Branch-and-Cut approach from Zhu et al. (2006), that uses valid inequalities, and the Constraint Programming approach from Schnell and Hartl (2017). There is a difference in favor of the Branch-and-Cut and Constraint Programming approaches concerning the found results, although the LB adapted matheuristic have obtained results with an average deviation of approximately 1% for the best solutions or the best-known solutions, with a competitive runtime.

The implemented matheuristic became close to the state-of-the-art to the exact methods hybridized with heuristic techniques. Additionally, it may be easily enhanced to generate other pseudo-cuts, different from those pseudo-cuts discussed in the current article. Furthermore, the used local search may also be changed to the Best Improvement Local Search heuristic or even to other metaheuristic structures.

Declaration of Interest, Compliance and Ethical Standards

Authors Gustavo Alves Fernandes and Sérgio Ricardo de Souza declare that they have no conflict of interest. This article does not contain any studies with human participants or animals performed by any of the authors.

Acknowledgements

The authors would like to thank the Coordination for the Improvement of Higher Education Personnel (CAPES), theMinas Gerais State Research Foundation (FAPEMIG), the National Council of Technological and Scientific Development (CNPq), and the Federal Center of Technological Education of Minas Gerais (CEFET-MG) for supporting the development of the present study. This study was financed in part by the Coordination for the Improvement of Higher Education Personnel (CAPES) - Brazil - Finance Code 001.

References

Acuna-Agost, R., Michelon, P., Feillet, D., & Gueye, S. (2011). A MIP-based local search method for the railway rescheduling problem. *Networks, 57*, 69–86.

Almeida, B. F., Correia, I., & Saldanha-da-Gama, F. (2019). Modeling frameworks for the multi-skill resource-constrained project scheduling problem: a theoretical and empirical comparison. *ITOR, 26*, 946–967. https://doi.org/10.1111/itor.12568

Asta, S., Karapetyan, D., Kheiri, A., Özcan, E., & Parkes, A. J. (2016). Combining monte-carlo and hyper-heuristic methods for the multi-mode resource-constrained multi-project scheduling problem. *Information Sciences, 373*, 476–498.

Bertsimas, D., & Tsitsiklis, J. N. (1997). *Introduction to Linear Optimization* (Vol. 6). MA: Athena Scientific Belmont.

Blazewicz, J. (1986). *Scheduling under resource constraints: Deterministic models* (Vol. 7). JC Baltzer.

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

Boctor, F. F. (1993). Heuristics for scheduling projects with resource restrictions and several resource-duration modes. *International Journal of Production Research, 31*, 2547–2558.

Boschetti, M. A., Maniezzo, V., Roffilli, M., & Bolufé Röhler, A. (2009). Matheuristics: Optimization, simulation and control. In M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, & A. Schaerf (Eds.), *Hybrid Metaheuristics* (pp. 171–177). Berlin, Heidelberg: Springer.

Bouleimen, K., & Lecocq, H. (2003). A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version. *European Journal of Operational Research, 149*, 268–281.

Chakrabortty, R. K., Abbasi, A., & Ryan, M. J. (2020). Multi-mode resource-constrained project scheduling using modified variable neighborhood search heuristic. *International Transactions in Operational Research, 27*, 138–167. https://doi.org/10.1111/itor.12644

Changchun, L., Xi, X., Canrong, Z., Qiang, W., & Li, Z. (2018). A column generation based distributed scheduling algorithm for multi-mode resource constrained project scheduling problem. *Computers & Industrial Engineering, 125*, 258–278.

Chiang, C.-W., Huang, Y.-Q., & Wang, W.-Y. (2008). Ant colony optimization with parameter adaptation for multi-mode resource-constrained project scheduling. *Journal of Intelligent & Fuzzy Systems: Applications in Engineering and Technology, 19*, 345–358.

Damak, N., Jarboui, B., Siarry, P., & Loukil, T. (2009). Differential evolution for solving multi-mode resource-constrained project scheduling problems. *Computers & Operations Research, 36*, 2653–2659.

Demeulemeester, E. L. (2002). *Project scheduling: a research handbook* (Vol. 102). Springer.

Fernandes, G. A., & de Souza, S. R. (2017). A CLONALG-inspired algorithm with adaptive large neighborhood for the multi-mode resource-constrained project scheduling problem. In *Proceedings of the 12th edition of the Metaheuristics International Conference (MIC 2017), Barcelona, July 4-7* (pp. 561–570). MIC/MAEB.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming, 98*, 23–47.

Fréville, A. (2004). The multidimensional 0–1 knapsack problem: An overview. *European Journal of Operational Research, 155*, 1–21. https://doi.org/10.1016/S0377-2217(03)00274-1

Geiger, M. J. (2017). A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. *European Journal of Operational Research, 256*, 729–741. https://doi.org/10.1016/j.ejor.2016.07.024

Gerhards, P., Stuerck, C., & Fink, A. (2017). An adaptive large neighbourhood search as a matheuristic for the multi-mode resource-constrained project scheduling problem. *European Journal of Industrial Engineering, 11*, 774–791.

Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research, 207*, 1–14.

Hartmann, S., & Briskorn, D. (2021). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*. https://doi.org/10.1016/j.ejor.2021.05.004

Hartmann, S., & Drexl, A. (1998). Project scheduling with multiple modes: a comparison of exact algorithms. *Networks, 32*, 283–297.

Hoos, H., & Stützle, T. (2004). *Stochastic Local Search: Foundations & Applications*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Kelley Jr, J. E., & Walker, M. R. (1959). Critical-path planning and scheduling. In *Papers presented at the December 1–3, 1959, eastern joint IRE-AIEE-ACM computer conference* (pp. 160–173). ACM.

Jarboui, B., Damak, N., Siarry, P., & Rebai, A. (2008). A combinatorial particle swarm optimization for solving multi-mode resource-constrained project scheduling problems. *Applied Mathematics and Computation, 195*, 299–308.

Kolisch, R. (1996). Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research, 90*, 320–333.

Kolisch, R., & Drexl, A. (1997). Local search for nonpreemptive multi-mode resource-constrained project scheduling. *IIE Transactions, 29*, 987–999.

Kolisch, R., & Sprecher, A. (1997). Psplib – A project scheduling problem library: Or software - orsep operations research software exchange program. *European Journal of Operational Research, 96*, 205–216.

Kolisch, R., Sprecher, A., & Drexl, A. (1995). Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science, 41*, 1693–1703. URL: http://www.jstor.org/stable/2632747.

Li, H., & Zhang, H. (2013). Ant colony optimization-based multi-mode scheduling under renewable and nonrenewable resource constraints. *Automation in Construction, 35*, 431–438.

Li, Z., Bookbinder, J. H., & Elhedhli, S. (2012). Optimal shipment decisions for an airfreight forwarder: Formulation and solution methods. *Transportation Research Part C: Emerging Technologies, 21*, 17–30.

Lodi, A. (2009). Mixed integer programming computation. In M. Jünger, T. M. Liebling, D. Naddef, G. L. Nemhauser, W. R. Pulleyblank, G. Reinelt, G. Rinaldi, & L. A. Wolsey (Eds.), *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-art* (pp. 619–642). Springer Science & Business Media.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L. P., Birattari, M., & Stützle, T. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives, 3*, 43–58.

Lova, A., et al. (2009). An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics, 117*, 302–316.

Mika, M., Waligora, G., & Weglarz, J. (2008). Tabu search for multi-mode resource-constrained project scheduling with schedule-dependent setup times. *European Journal of Operational Research, 187*, 1238–1250.

Mitchell, J. E. (2009). Integer programming: Branch and cut algorithms. In C. A. Floudas, & P. M. Pardalos (Eds.), *Encyclopedia of Optimization*, 2nd ed. (pp. 1643–1650). Springer. doi: 10.1007/978-0-387-74759-0_287.

Muller, L. F. (2011). An Adaptive Large Neighborhood Search Algorithm for the Multi-mode RCPSP. Report 3 Department of Management Engineering, Technical University of Denmark. Kgs. Lyngby, Denmark.

Muritiba, A. E. F., Rodrigues, C. D., & da Costa, F. A. (2018). A path-relinking algorithm for the multi-mode resource-constrained project scheduling problem. *Computers & Operations Research, 92*, 145–154.

Noori, S., & Taghizadeh, K. (2018). Multi-mode resource constrained project scheduling problem: A survey of variants, extensions, and methods. *International Journal of Industrial Engineering & Production Research, 29*, 293–320. https://doi.org/10.22068/ijiepr.29.3.293

Okada, I., Takahashi, K., Zhang, W., Zhang, X., Yang, H., & Fujimura, S. (2014). A genetic algorithm with local search using activity list characteristics for solving resource-constrained project scheduling problem with multiple modes. *IEEJ Transactions on Electrical and Electronic Engineering, 9*, 190–199.

Patterson, J., Slowinski, R., Talbot, F., & Weglarz, J. (1989). An algorithm for a general class of precedence and resource constrained scheduling problems. In R. J. Weglarz (Eds.), *Advances in Project Scheduling* (pp. 3–28). Elsevier.

Raidl, G. R., & Puchinger, J. (2008). Combining (integer) linear programming techniques and metaheuristics for combinatorial optimization. In C. Blum, M. J. B. Aguilera, A. Roli, & M. Sampels (Eds.), *Hybrid Metaheuristics: An Emerging Approach to Optimization* (pp. 31–62). Berlin, Heidelberg: Springer. https://doi.org/10.1007/978-3-540-78295-7_2.

Rodríguez-Martín, I., & Salazar-González, J. J. (2010). A local branching heuristic for the capacitated fixed-charge network design problem. *Computers & Operations Research, 37*, 575–581.

Santos, H. G., Soares, J., & Tóffolo, T. (2014). Hybrid local search for the multi-mode resource-constrained multi-project scheduling problem. In *Proceedings of the 10th International Conference on the Practice and Theory of Automated Timetabling (PATAT'14)* (pp. 397–407).

Schnell, A., & Hartl, R. F. (2017). On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs. *Operations Research Perspectives, 4*, 1–11.

Sprecher, A. (1994). Resource-constrained project scheduling: Exact methods for the multi-mode case. In *Lecture Notes in Economics and Mathematics*.

Sprecher, A., & Drexl, A. (1998). Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research, 107*, 431–450.

Sprecher, A., Hartmann, S., & Drexl, A. (1997). An exact algorithm for project scheduling with multiple modes. *Operations-Research-Spektrum, 19*, 195–203.

Talbot, F. B. (1982). Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science, 28*, 1197–1210.

Van Peteghem, V., & Vanhoucke, M. (2009). An artificial immune system for the multi-mode resource-constrained project scheduling problem. In *Evolutionary computation in combinatorial optimization* (pp. 85–96). Springer.

Van Peteghem, V., & Vanhoucke, M. (2010). A genetic algorithm for the preemptive and non-preemptive multi-mode resource-constrained project scheduling problem. *European Journal of Operational Research, 201*, 409–418.

Van Peteghem, V., & Vanhoucke, M. (2011). Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. *Journal of Heuristics, 17*, 705–728.

Van Peteghem, V., & Vanhoucke, M. (2014). An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. *European Journal of Operational Research, 235*, 62–72.

Wauters, T., Kinable, J., Smet, P., Vancroonenburg, W., Berghe, G. V., & Verstichel, J. (2016). The multi-mode resource-constrained multi-project scheduling problem. *Journal of Scheduling, 19*, 271–283.

Wauters, T., Verbeeck, K., Berghe, G. V., & De Causmaecker, P. (2011). Learning agents for the multi-mode project scheduling problem. *Journal of the Operational Research Society, 62*, 281–290. https://doi.org/10.1057/jors.2010.101

Weglarz, J., Józefowska, J., Mika, M., & Waligóra, G. (2011). Project scheduling with finite or infinite number of activity processing modes – A survey. *European Journal of Operational Research, 208*, 177–205. https://doi.org/10.1016/j.ejor.2010.03.037

Zhang, H., Tam, C., & Li, H. (2006). Multimode project scheduling based on particle swarm optimization. *Computer-Aided Civil and Infrastructure Engineering, 21*, 93–103.

Zhu, G., Bard, J. F., & Yu, G. (2006). A branch-and-cut procedure for the multimode resource-constrained project-scheduling problem. *INFORMS Journal on Computing, 18*, 377–390.