European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Invited Review in Celebration of the 50th Anniversary of EURO

Fifty years of research on resource-constrained project scheduling explored from different perspectives

Christian Artigues [a,*], Sönke Hartmann [b,*], Mario Vanhoucke [c,d,e,**]

[a] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[b] HSBA Hamburg School of Business Administration, Willy-Brandt-Straße 75, D-20459, Hamburg, Germany
[c] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000, Gent, Belgium
[d] Technology and Operations Management, Vlerick Business School, Reep 1, 9000, Gent, Belgium
[e] UCL School of Management, University College London, 1 Canada Square, London, E14 5AA, United Kingdom

ARTICLE INFO

Keywords:
Project scheduling
Resource-constraints
Literature review
Project data
Methodological advances
Variants and extensions

ABSTRACT

The resource-constrained project scheduling problem is one of the most investigated problems in the project scheduling literature, and has a rich history. This article provides a perspective on this challenging scheduling problem, without having the ambition to provide a complete overview. Instead, the article does aim to summarize a number of reasons why this problem has been so intensely investigated from different perspectives.

It will be shown that this scheduling problem has many faces, and therefore deserves a lot of research time from a computational and theoretical point of view as well as from a practical point of view. An overview of possible extensions to other problems and a detailed overview of the used (both heuristic and exact) solution methods will be given. In addition, the data used will be discussed and interesting avenues for further research will be mentioned throughout the different sections.

1. Introduction and problem definition

The problem of assigning start times to activities of a project subject to precedence constraints in order to minimize the project total duration was the subject of intensive research in the 1950's because of its relevance to large-scale industrial or military projects. Researchers found that the problem can be solved in polynomial time by finding longest paths, or equivalently by duality minimum potentials, in the precedence graph. Efficient methods were independently proposed such as the program evaluation and review technique (PERT) (Malcolm et al., 1959), the critical path method (Kelley Jr & Walker, 1959) and the metra potential method (Roy & Dibon, 1966). Incorporating resource constraints in this problem gave birth in the 1960's to the *resource-constrained project scheduling problem* (RCPSP) (Dike, 1964; Kelley, 1963; Wiest, 1964) and a first survey appeared in 1966 (Davis, 1966). During this decade and in the 1970's, complexity results were established and the first integer programming formulations and heuristic methods dealing with resource-constraints were proposed. In his survey, Herroelen (1972) mentions already 54 papers, and the number of new studies has grown exponentially since then. The interest on the RCPSP and its variants grew rapidly in the 1980's with different methods to solve this problem under multiple conflicting criteria (Slowinski, 1980) and different categories of resources (Slowinski, 1981). From the 1990's on, the research studies started with the development of various heuristic and metaheuristic approaches, powerful dedicated branch-and-bound methods based on specific dominance rules, lower bounds and a large research effort on algorithm competition on various problem instance libraries. This growth is well illustrated by publications at that time of several survey studies (Brucker et al., 1999; Herroelen et al., 1998; Icmeli et al., 1993; Kolisch & Padman, 2001; Özdamar & Ulusoy, 1995) with references to 42, 83, 22, 203 and 211 papers, respectively. The next two decades saw the advent of constraint programming, satisfiability problem (SAT) based and newly efficient integer programming methods as well as more and more sophisticated metaheuristics and hybridizations. The maturity of the field is assessed by the publication of several books (Artigues et al., 2013; Brucker & Knust, 2012; Demeulemeester & Herroelen, 2006; Dorndorf, 2002; Neumann et al., 2002; Schwindt et al., 2015; Vanhoucke, 2012). This

* Corresponding authors.
** Corresponding author at: Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000, Gent, Belgium.
E-mail addresses: artigues@laas.fr (C. Artigues), Soenke.Hartmann@hsba.de (S. Hartmann), Mario.Vanhoucke@UGent.be (M. Vanhoucke).
URLs: https://www.laas.fr/fr/homepages/artigues/ (C. Artigues), https://www.hsba.de/hartmann (S. Hartmann), https://www.vlerick.com/en/find-faculty-and-experts/mario-vanhoucke/ (M. Vanhoucke).

https://doi.org/10.1016/j.ejor.2025.03.024
Received 26 December 2023; Accepted 23 March 2025
Available online 26 April 2025

paper reviews some of the most investigated research lines on the RCPSP and highlights some reasons of this never-ending enthusiasm for this challenging resource-constrained scheduling problem.

First and foremost, researchers spend their time on developing new procedures because the problem has an impressive variety of applications. Initially and still today, the RCPSP is a combinatorial optimization problem situated at the core of project management (Demeulemeester & Herroelen, 2006; Vanhoucke, 2012). The problem is proposed as the basic problem in the project scheduling literature and follows a number of assumptions that, despite the fact that they often do not arise in this way in practice, make the problem attractive for academic research. Naturally, the complexity of the problem is one of the main reasons why researchers continuously develop improved scheduling algorithms. However, one of the most attractive aspects of the well-defined assumptions of the RCPSP is that many real project scheduling problems that arise in practice often contain the basic problem as a sub-problem. These assumptions, which in practice are often regarded as too restrictive and insufficiently applicable for real projects, can therefore also be seen as an advantage because it allows to define many extensions in a simple manner without affecting the core of the scheduling problem. The best example of such a modular approach can be seen in the classification schemes of Brucker et al. (1999) and Herroelen et al. (1999) that have classified the basic problem and its various possible extended components, in order to provide a clear picture of the many applications of this problem. As an example, in a study by Vanhoucke (2013), a number of possible extensions to the basic RCPSP formulation was tested on a number of real projects from practice to demonstrate that the problem, if properly defined, does indeed have many applications. It is also no coincidence that thanks to its practical value, this problem was also included in most commercial software tools.

The problem appears in many other and sometimes unexpected application domains such as make-to-order production scheduling (Wang et al., 2021), batch scheduling in the process industry (Schwindt & Trautmann, 2000), assembly line balancing (De Reyck & Herroelen, 1995), timetabling (Brucker & Knust, 2000b), evacuation planning (Even et al., 2015), hazardous material inspection scheduling (Polo-Mejía et al., 2023), spacecraft experiments scheduling (Simonin et al., 2015), register allocation and instruction scheduling in compilers (Lozano & Schulte, 2019), circuit generation by high level synthesis (Sittel et al., 2018). Moreover, the RCPSP is extremely computationally challenging even on problem instance of modest size, even with the tremendous progress made in the 1990s for solving disjunctive scheduling problems (a particular case of the RCPSP where any resource can perform only one activity at a time). Notably, there are still today 60-activity RCPSP instances resisting to the exact solvers since almost 30 years.

In what follows, we formally define the RCPSP and discuss its position in the scheduling literature that may explain its current intractability. Then, we present the different sections of this survey. The problem considers a set of activities $\mathcal{A}$ constituting the project, a set of resources $\mathcal{R}$ and a time horizon $\mathcal{T} = [0, T)$, where T is an upper bound on the project duration. Each resource $k \in \mathcal{R}$ has a constant integer, non negative availability B_k. The resources are of the renewable type, meaning that once an activity that required the resource is completed, the resource units occupied by the activity are released and can be used again by another activity. A resource may model various real-life resources such as a team of technicians of identical skills, an area with limited space, etc. An activity, also called a task or an operation in other contexts where the RCPSP model applies, is an elementary stage of the project with a uniform resource usage. Each activity $i \in \mathcal{A}$ has a constant integer processing time p_i and requires a non negative, also constant and integer, amount b_{ik} of each of the resources $k \in \mathcal{R}$. A typical example of an activity in aircraft assembly is a riveting operation on the fuselage which requires 2 technicians, a surface area of 5 m^2 in the assembly area and lasts 2 h. The problem consists in assigning a start time S_i to each activity, which inductively defines its completion time C_i equal to the start time S_i plus the processing time p_i so that at any time point, the resource usage of the activities in progress never exceeds the resource availability, which is called the cumulative constraint.[1] For example, if we have a team of only 5 technicians then only two riveting activities can be in progress simultaneously. Assuming that the area around the aircraft is subdivided in 5 m^2 zones, among two riveting activities located in the same zone, only a single one can be executed at any time period due to space limitation. The activities are also linked by precedence constraints that are usually represented by means of an activity-on-node directed graph with arc set E where an arc $(i, j) \in E$ from an predecessor activity $i \in \mathcal{A}$ to a successor activity $j \in \mathcal{A}$ models the fact that the successor activity cannot be started while the predecessor activity is not completed. Coming back to the aircraft assembly example, the fuselage elements must be positioned before the riveting activity can take place, which is modeled by a precedence constraint linking the corresponding positioning and riveting activities. The most common objective is to minimize the total project duration or makespan $C_{\max}$.

The RCPSP can be formally stated as follows:

$$
\begin{aligned}
\min \quad & C_{\max} \\
\text{s.t.} \quad & C_{\max} \geq S_i + p_i && \forall i \in \mathcal{A}, \\
& S_j \geq S_i + p_i && \forall (i,j) \in E, \\
& \sum_{i \in \mathcal{A}(t)} b_{ik} \leq B_k && \forall k \in \mathcal{R}, t \in \mathcal{T}
\end{aligned}
$$

where $\mathcal{A}(t)$ gives the subset of activities in progress at time point t, i.e. such that $t \in [S_i, S_i + p_i[$. The objective and the first constraints set the makespan as the maximum completion time among all activities. The second constraints are the precedence constraints and the last constraints are the cumulative constraints.

The RCPSP occupies a special place in what is now humorously called the *scheduling zoo* (Dürr, 2023), which we can define as the (huge) family of different scheduling problems that can be obtained by varying the parameters of the famous three-field notation of the form $\alpha|\beta|\gamma$ where α describes the machine environment (1 for single machine, P for parallel machine, F for flow-shop, O for open-shop, J for job-shop, etc.), β gives additional constraints (such as precedence constraints denoted by $prec$, release dates are denoted by r_i, etc.) and γ gives the objective function (e.g. $C_{\max}$). In 1979, when (Graham et al., 1979) introduced the three-field notation, the RCPSP could be denoted as $P\infty|prec, res|C_{\max}$, where res indicates the presence of renewable resources of limited capacity (precisely the standard resource constraints as defined above in the formal RCPSP model). To have the number of activities simultaneously scheduled in parallel only limited by these resource constraints, the machine environment had to be relaxed to an infinite number of parallel machines, which is denoted by $\alpha = P\infty$.[2] More notation schemes able to classify variants and extensions of the RCPSP were proposed by the previously mentioned classifications studies of Herroelen et al. (1998) where the RCPSP is denoted $m, 1|cpm|C_{\max}$ and Brucker et al. (1999) where the RCPSP is denoted $PS|prec|C_{\max}$. Discussions about the merits and drawbacks of these classification schemes can be found in Herroelen et al. (2001). In what follows, we will use the Graham et al. (1979) notation, which is sufficient for the standard RCPSP and its special cases.

[1] In the constraint programming literature, such renewable resources on which the cumulated resource usage at each time point cannot exceed the resource capacity, are called *cumulative* resources and the corresponding constraint is called the *cumulative* constraint (see e.g. Schutt et al., 2013), while other authors use sometimes the *cumulative* term for storage resources (Schwindt & Trautmann, 2000).

[2] In the three-field notation, $\alpha = Pm$ means a number of machines fixed to m while $\alpha = P$ means an arbitrary number of machines, which is part of the input.

Despite the simplicity of its definition, the RCPSP offers a rich modeling framework as it admits several well-known scheduling problems as special cases. For example, if each activity $i \in \mathcal{A}$ has an individual due date d_i and if instead of the makespan, one wants to minimize the maximum lateness $L_{\max} = \max_{i\in\mathcal{A}} S_i + p_i - d_i$ then it suffices to consider an RCPSP instance with activity set $\mathcal{A}'$, precedence constraint set E' and the same resource set $\mathcal{R}$ such that all activities of $\mathcal{A}$ are present in $\mathcal{A}'$ with the same characteristics. In addition, for each activity $i \in \mathcal{A}$, an activity i' is present in $\mathcal{A}'$ of duration $D - d_i$, no resource demand and having i as unique predecessor. D must be a constant larger than $\max_{i\in\mathcal{A}} d_i$ for having non negative processing times. Hence, precedence constraint set E' includes all precedence constraints in E plus an arc (i, i') for each activity $i \in \mathcal{A}$. Any method that minimizes the makespan in the RCPSP instance built this way minimizes the maximum lateness in the original instance.

If each activity $i \in \mathcal{A}$ has additionally a release date r_i yielding constraint $S_i \geq r_i$, a similar instance modification can be made to incorporate all release dates. Each activity i is paired with an additional predecessor having a processing time equal to r_i and no resource demand.

We just saw the RCPSP model can be used to solve the more general $P\infty|prec, res, r_i|L_{\max}$ problem, involving release dates, due date and the maximum lateness criterion. Taking this equivalence into account and by varying the resource availability the activity requirements and the precedence constraint form, the RCPSP also encompasses the famous one-machine problem $1|r_i|L_{\max}$, several parallel machine problems such as $P|r_i, prec|L_{\max}$, the flow-shop problem $F \parallel C_{\max}$, the open-shop problem $O \parallel C_{\max}$, the job-shop problem $J \parallel C_{\max}$ and some flexible variants of the last two problems.

The fact that the RCPSP has all these NP-hard problems as particular cases inspires at least two reflexions. First, the RCPSP is obviously an NP-hard optimization problem. However, the complexity status of simpler particular cases has been studied and will be reviewed in Section 2. Second, the RCPSP structure is highly polymorphic. Knowing the gigantic amount of research papers that have been necessary before obtaining efficient methods to solve, exactly or approximatively the job-shop problem, to name only this one, having a generic and stable method to solve (nearly) optimally the RCPSP in all its generality appears a more than serious challenge. This is why the RCPSP is the ideal playground for all categories of combinatorial optimization methods. Section 3 discusses the different data generators and benchmark instance sets that have been produced to evaluate these methods. Concerning exact solution approaches Sections 4–6 review the mixed-integer linear programming methods, constraint programming and SAT-based methods and specific branch-and-bound methods, respectively. Section 7 is devoted to heuristics and metaheuristics. Section 8 presents the different lower bounds that have been investigated. The numerous mentioned real-life applications of the RCPSP often involve additional constraints and/or objectives and optimization contexts than the standard RCPSP. Section 9 briefly reviews the different variants of the RCPSP while Section 10 draws some final conclusions and highlights some future research paths.

2. Structural properties and complexity

Several observations can be made from the formulation given in Section 1. First, despite the fact that the start time variables are continuous, there always exist an integer optimal solution if all data parameters also take integer values (Artigues, 2008). Hence, a restriction to integer start times can be assumed. Second, omitting resource constraints yield the standard resource-unconstrained project scheduling problem that can be solved in polynomial time since the formulation of Section 1 is a linear program with $|\mathcal{A}|$ continuous variables and $|E|+|\mathcal{A}|$ constraints. The problem can be nicely represented and efficiently solved via the activity-on-node graph representation. A graph is build with a node per activity including a dummy start activity 0 and a dummy end activity $|\mathcal{A}| + 1$ both of zero duration. An arc is defined between two node associated to activities linked by a precedence constraint, considering that activity 0 is a predecessor of all other nodes while node $|\mathcal{A}| + 1$ is a successor of all other nodes. The arc is valuated by the duration of the origin activity. Then, the minimum makespan of the problem is equal to the length of the longest path in this graph from 0 to $|\mathcal{A}| + 1$. This is the core of the PERT, CPM and MPM methods mentioned above. It follows that the problem difficulty comes from the adjunction of resource constraints.

2.1. Complexity analysis and approximation

A fundamental question is to establish the boundary between polynomially solvable problems and NP-hard particular cases of the RCPSP. One of the first such results was established by Lenstra and Rinnooy Kan (1978) who found minimal NP-hardness results when adding basic resource constraints: all activities require one machine to be selected among two identical ones. They proved that while $P2|prec, p_i = 1|C_{\max}$ is polynomially solvable, $P2|prec, p_i \in \{1, 2\}|C_{\max}$ is strongly NP-hard by reduction from Clique.[3] They also established that $P|tree, p_i = 1|C_{\max}$ (the precedence constraints form a tree) can be solved in polynomial time whereas $P|prec, p_i = 1|C_{\max}$ is NP-hard, also by reduction from Clique. This illustrates the influence of the processing time variability and the precedence constraint structure on the complexity.

Another question getting closer to the RCPSP is to determine the complexity boundary when adding renewable resource constraints to simple parallel machine problems with or without precedence constraints. Under the Graham et al. (1979) notation, Garey and Johnson (1975) showed that the unit duration RCPSP with two parallel machines $P2|res, p_i = 1|C_{\max}$ is polynomially solvable while $P3|res1, p_i = 1|C_{\max}$ and $P2|res1, tree, p_i = 1|C_{\max}$ are both NP-hard, where $res1$ stands for the consideration of a unique renewable resource ($|\mathcal{R}| = 1$). Blazewicz et al. (1983) further refined their boundaries by considering other particular cases identified by the notation $res\delta\sigma\rho$ where δ is the number of resources, σ is an upper bound on the maximal resource availability and ρ is an upper bound on the resource requirement of each activity on any resource. They showed that $P3|res.11, p_i = 1|C_{\max}$ and that $P2|res111, chain, p_i = 1|C_{\max}$ are both strongly NP-hard, where $chain$ means that precedence constraints are in the form of a set of chains of activities. The source of difficulty that these results reveal is not related to the cumulative constraint (i.e. the presence of requirements on a resource available in multiple units) since $\sigma = 1$ and tasks have unit processing times, but rather to the fact that the activities require several resources simultaneously for the first case and that the number of required resources can vary from 0 to 1, jointly with precedence constraints, even in the form of chains, in the second case.

The multi-resource source of difficulty can be also illustrated by reduction from the graph coloring problem for unit time activities $P\infty|res.11, p_i = 1|C_{\max}$: an RCPSP instance is obtained by defining a resource for each arc and the makespan corresponds to the number of colors (Schäffter, 1997). On the other hand, about the influence of cumulative constraint on the problem difficulty, note that the bin packing problem is a famous special case of the RCPSP that can be denoted $P\infty|res1.., p_i = 1|C_{\max}$. It can be considered as the simplest cumulative scheduling problem and is known to be strongly NP-hard (Garey & Johnson, 1979).

On the positive side, polynomial-time complexity results can be obtained by fixing a part of the input. Blazewicz and Ecker (1983) showed that when all resource parameters (number of resources, maximal capacity and maximal requirement) are fixed, problem $P\infty|res\delta\sigma\rho, p_i = 1|C_{\max}$ can be solved in linear time. Blazewicz and Kubiak (1989) considered the case of typed activities where all tasks of the same type have

[3] Clique is the NP-complete problem answering whether a clique of a given cardinality exists in an undirected graph.

the same processing time and resource requirements. When the number of activity classes is fixed and either the number of parallel machines or the maximum processing time is also fixed ($P|res, p_i < p, types = k|C_{\max}$ and $Pm|res, types = k|C_{\max}$), Blazewicz and Kubiak (1989) proposed a polynomial time dynamic programming algorithm. Brucker and Krämer (1996) extended these results to $P|res, p_i < p, types = k, r_i|C_{\max}$. Kovalyov and Shafransky (1998) provided an $O(1)$ algorithm for $P|res1.1, p_i = 1|C_{\max}$. A synthesis of known complexity results for the RCPSP, i.e. the complexity map of the problem, can be found in Ganian et al. (2020).

For some of the above-considered NP-hard RCPSP special cases, there exists polynomial time approximation algorithms with performance guarantee. First, for unit duration activities, Graham et al. (1979) showed that for $P\infty|res, prec, p_i = 1|C_{\max}$ the algorithm that sorts the activities in a non-decreasing order of maximum resource requirements (DMR) has a worst case approximation ratio of $\frac{17}{10}|\mathcal{R}|+1$. Without precedence constraints ($P\infty|res, p_i = 1|C_{\max}$), the makespan produced by the list scheduling[4] with an arbitrary priority list is at most $|\mathcal{R}| + \frac{7}{10}$ times the optimal value, plus a constant factor of $\frac{7}{2}$. For the bin-packing special case, a survey of approximation results can be found in Coffman et al. (2013). Restricting the problem to a single resource and m parallel machines, for $P|res1, p_i = 1|C_{\max}$, the makespan of the solution produced by the DMR algorithm is at most $2 - \frac{2}{m}$ times the optimal makespan plus a constant factor of 1 (Krause et al., 1975). Restricting further the activities to occupy 0% or 100% of the resource ($P|res111, p_i = 1|C_{\max}$), Goyal (1976) showed that the worst case ratio of list scheduling is $3 - \frac{2}{m}$. For the more general case $P|res, prec|C_{\max}$, i.e. the RCPSP in which no more than m jobs can be scheduled in parallel, list scheduling has a worst case ratio of m (Garey & Graham, 1975). Without precedence constraints ($P|res|C_{\max}$), the worst case ratio of list scheduling becomes $\min(\frac{m+1}{2}, s+2-\frac{2s+1}{m})$, with $s = |\mathcal{R}|$. Without the m-machine restriction, $P\infty|res|C_{\max}$, the ratio of list scheduling improves to $|\mathcal{R}| + 1$. A survey on the complexity of parallel machine problems with resource constraints can be found in Edis et al. (2013). However more negative approximation results were established. Because of the equivalence with graph coloring, the RCPSP is not in APX, which define the class of problems having a worst-case ratio bounded by a constant (Uetz, 2001). More recently, Gafarov et al. (2014) provided non approximability results for less specific instances.

Research on the complexity of combinatorial optimization problems gained recently a new interest with parameterized complexity. A problem is fixed parameter tractable (FPT) if one can find an optimal algorithm that runs in a polynomial time w.r.t. the input size, but in a potentially exponential time w.r.t. a given parameter (Downey & Fellows, 2012). Bevern et al. (2016) proved that the RCPSP is fixed-parameter tractable parameterized with the width of the precedence graph combined with the maximum difference between the earliest start time and the factual start time of any activity.

2.2. Structural properties

Given the NP-hardness of the RCPSP, structural properties have been established to reduce the search space. Indeed, even if the activity start times can be restricted to integer values, in a problem without precedence constraints there are $\Pi_{i\in A}(T - p_i)$ different start time vectors, which yields a huge search space. Bartusch et al. (1988) study an order theoretic model for the RCPSP inspired by the disjunctive graph approach for machine scheduling (Balas, 1969) as well as by the resource-unconstrained case and the potentials in the precedence graphs evoked in the introduction. They established that, geometrically, the solution space of the RCPSP is a union of finitely many polyhedra, each polyhedron defining the set of potentials associated to the precedence graph augmented by additional precedence constraints given by a so called *feasible strict partial order*.[5] The rationale is the intuition that if the earliest start schedule associated to the precedence graph is infeasible, then there is a set of activities in process simultaneously that violates the resource constraint because on some resource the sum of their requirement exceeds the resource capacity. Such a set is called a *forbidden set*. At least one precedence constraint must be added between two activities of each forbidden set to obtain a feasible schedule. So a strict partial order is feasible if the precedence constraints it adds to the precedence graph destroys all inclusion-minimal forbidden sets (MFS). The RCPSP then resorts to finding the feasible strict partial order yielding the precedence graph of minimal longest path length. The set of earliest start schedules associated with a feasible strict partial order is called the set of quasi-active schedule (Neumann et al., 2000). It is strictly included in the feasible set and dominant for makespan minimization. Unfortunately, the number of feasible strict partial orders can be huge.

Beyond the computational complexity gap from the resource-uncon strained to the resource-constrained case illustrated in the previous section, this definition of the RCPSP underlines that some intuitive properties have to be reconsidered when limited resources are introduced. The typical example is that of the activity slack time and criticality. Without resource constraint, the slack of an activity is the difference between the longest path length in the precedence graph and the length of the longest path passing through the activity. It gives the time interval on which the activity can start without increasing the minimum project duration and critical activities, the ones with a slack equal to zero, should be monitored carefully as any increase on their start time unavoidably increase the end time of the project. Now with resource constraints, given an optimal set of additional precedence constraints destroying all forbidden sets, we can define slacks and critical activities as in the resource-unconstrained case. But there exist in general may optimal ways of adding precedence constraints, each potentially leading to different slacks and critical activities. Hence, as already remarked by Wiest (1964), slacks and criticality are ill-defined and so misleading concepts in the RCPSP. This questions the use of methods based on activity slacks such as the critical chain methods (Goldratt, 2017) whose aim is to insert buffers based on activity criticality. Merits and pitfalls of the critical chain method where analyzed by Herroelen and Leus (2001).

The search space can be further reduced by remarking that from a quasi-active schedule, left-shifting an activity by one unit while leaving the other activities at their start time may be feasible and cannot increase the makespan, as remarked by Sprecher et al. (1995). Such a shift is called a local left shift by Kolisch (2015) and is equivalent for the RCPSP to the order-monotonic left shift presented by Neumann et al. (2000). The set of schedules in which no feasible local left shift exists is called the set of semi-active schedules and corresponds to the set of pseudo-active schedules given by Neumann et al. (2000). This set is in turn included in the set of quasi-active schedules.

The smallest set of dominant schedules considered in the literature is the set of active schedules, for which no global left-shift (a left shift of more than one unit) exists. Interestingly, this set can be generated by enumerating all activity permutations (total orders) compatible with the precedence constraints and computing for each of them a schedule by the serial generation scheme algorithm (Kolisch, 2015) that takes the activities one by one in the permutation order and schedule each activity as early as possible. This method is widely used in heuristic approaches (see Section 7). The search space reduction can be drastic, as for 4 activities without precedence constraints, there are 219 possible strict partial orders and only 24 permutations. Even if these

[4] The list scheduling algorithm in presence of resource constraints is similar to the parallel scheduling scheme described in Section 7.

[5] In mathematics a strict partial order in an irreflexive, antisymmetric and transitive binary relation on a set.

numbers are only upper bounds on the numbers of pseudo-active and active schedules, respectively, as different orders may give the same schedule, this give an idea of the drastic reduction on the search space that considering active schedules can bring. However, generating only active schedules in a branch-and-bound scheme is not trivial and some efficient exact methods for the RCPSP work in the quasi-active schedule space, such as the one by Laborie (2005).

3. Project data

Despite the known complexity of the RCPSP, enormous progress has been made in the last few decades in the use of data. The research took off when Patterson (1984) introduced his well-known dataset consisting of 110 instances extracted from various papers. This dataset soon became the standard to perform experiments on, until it was shown in 1992 that all instances could be optimally solved (using the algorithm of Demeulemeester & Herroelen, 1992). Kolisch and Sprecher (1996) therefore decided to generate a new project scheduling library (PSPLIB) that in no time became (and still is) the standard dataset on which to test new algorithms. Despite the fact that many of these instances already have a known optimal solution, quite a few of these PSPLIB instances are not solved to optimally to this day, and it can be expected that, even with the current rapid progress in research, this set will last for some time to come.

Unlike the Patterson set, the PSPLIB set does not consist of a random collection of projects, but they were accurately generated with a generator that uses network and resource indicators to control the generation process. This network generator ProGen (Kolisch et al., 1995) uses the *coefficient of network complexity* (CNC, Pascoe, 1966) to control network structure, and the *resource strength* (RS, Kolisch et al., 1995) to determine resource use. Given the success of the dataset, a number of other researchers started to build their own alternative network generator, or proposed extensions and improvements to the indicators. New network generators emerged such as ProGen/Max (Schwindt, 1995), RanGen (Demeulemeester et al., 2003; Vanhoucke et al., 2008) and RiskNet (Tavares, 1999). They relied on other network indicators such as *order strength* (OS, Mastor, 1970), the *serial/parallel indicator* (SP, Vanhoucke et al., 2008) and other resource indicators such as *resource-constrainedness* (RC, Patterson, 1976) to generate the projects. This growth of new generators also very quickly led to new datasets and we would like to refer the reader to two websites where the data (and solutions) for the RCPSP were made available.[6] Much of the data available at these websites is the results of various research studies for which a summary is recently provided by Vanhoucke (2024).

Elmaghraby and Herroelen (1980) were among the first to draw our attention to the importance of estimating the complexity of the project scheduling problem with indicators, but their real importance was only fully emphasized when Herroelen and De Reyck (1999) discussed the presence of so-called *phase transitions* for the RCPSP. A phase transition shows a sudden change in complexity for the problem, where one algorithm can handle it better than the other. The ultimate goal of such a study is to provide insights into the threshold values for the indicators that indicate such a transition, so that future algorithms can better respond to them. Tailoring (exact or heuristic) scheduling algorithms by taking into account specific values for the instance indicators is often done, but should done be much better in the future. After all, a number of researchers have shown that the existing network and resource indicators are not always reliable for predicting this complexity, and that results from different studies contradict each other. Vanhoucke and Coelho (2021) have shown that the resource indicator values can even have different values for the same instance, which can completely obscure our knowledge about the phase transition values. To overcome this problem, new complexity indicators (Van Eynde et al., 2024) and a new theoretical framework (Van Eynde & Vanhoucke, 2022b) to better discriminate between easy and hard instances for the RCPSP have been proposed. In addition to new complexity indicators, new artificial project datasets were recently generated, leading to the NetRes dataset (Vanhoucke & Coelho, 2018) for machine learning training purposes, the CV set (Coelho & Vanhoucke, 2020) for testing exact algorithms, and the sD set (Coelho & Vanhoucke, 2023) for testing meta-heuristics to solve the RCPSP.

What the future research to project data will bring is of course unknown, but there is little doubt that the new datasets and improved indicators can lead to new insights in several ways. A potentially promising future research avenue in the research of the RCPSP is the collection and use of empirical data, although this path is not yet very well explored. Perhaps the theoretical nature of the research has something to do with it, but what certainly also plays a role is that the basic assumptions of the RCPSP are not often met for real projects, which makes collecting real data not easy. For the time being, to the best of our knowledge, the empirical dataset of Batselier and Vanhoucke (2015) is the only freely available set that can be used for the research, although not all projects contain the necessary resource data. In contrast, many of these projects contain data on the risk profile of activities and sometimes even follow-up data reflecting the progress of the project, so these project can be used for different variants of the RCPSP (for realistic extensions of the basic assumptions of the RCPSP, see Section 9). At the time of the publication of the original paper, the data set contained 52 projects, but it already consisted of 183 projects at the time of publication of the current paper (and the set is still growing, cf. Vanhoucke, 2023). If the research community wants to make this challenging project scheduling problem and all its algorithms more practical, then testing on empirical data will be necessary to convince professional project managers that the gap between theory and practice is not unbridgeable.

One of the methods that can help to achieve this ambitious goal is a procedure to calibrate empirical project data. This idea was initially proposed in a paper by Trietsch et al. (2012) and involves a series of hypotheses tests using empirical progress data for activity durations. The calibration aims making these activity durations suitable for further research by testing whether they correspond to a predefined statistical distribution (they use the lognormal distribution as their null hypothesis). The novelty of their calibration procedure is that it cannot be reduced to a simple curve fitting method, since it incorporates the removal of data points that cannot be fully trusted due to being collected by humans (empirical data is collected by real project managers and not automatically generated by the network generators). The calibration procedure was validated on 24 projects from the empirical database by Colin and Vanhoucke (2016) and an extended version of it was extensively and successfully tested on 83 projects by Vanhoucke and Batselier (2019a, 2019b). Despite the fact that such research is still in its infancy, it may serve as an impetus to make the RCPSP more accessible to professional project managers, and it may also provide leverage in research into the stochastic version of this problem. It is therefore quite clear that research into project data for the RCPSP and all its extensions is still not over, and perhaps in the coming years, given the growing attention to data in general, many innovative ideas will be developed which (hopefully) will further increase our understanding of this complex scheduling problem.

4. Mixed-integer linear programming approaches

The brief historical review in the introduction reveals that, like for many other combinatorial optimization problems, mixed-integer linear

[6] The PSPLIB website to download instances and upload solutions is available at http://www.om-db.wi.tum.de/psplib/library.html. A similar system has been introduced for the single- and multi-mode version of the RCPSP at http://solutionsupdate.ugent.be. The most complete set with many instances and solutions for the RCPSP and many variants is available at https://www.projectmanagement.ugent.be/research/data.

programming branch and bound has most probably been the first exact method proposed for the RCPSP. Three types of MILP formulations are encountered in the literature, that can be distinguished by the type of integer variables they involve. There is a relation between these formulation types and the dominant schedule sets presented in Section 2. Surveys on MILP formulations for the RCPSP have been proposed by Artigues et al. (2015), Demassey (2013), Demeulemeester and Herroelen (2006). To compare different formulations, we are interested in their size, in terms of number of variables and/or constraints and their relaxation strength, in terms of the quality of the lower bound obtained by relaxing the integrity of the integer variables. More precisely, if P denotes a MILP formulation and $\mathcal{P}(P)$ denotes the convex polyhedron of its continuous relaxation, then a formulation P strictly dominates formulation P' if $\mathcal{P}(P) \subsetneq \mathcal{P}(P')$.

Weak and strong time-indexed formulations The most common MILP formulation is the time-indexed formulation aiming at representing all the feasible integer schedules by means of a binary variable x_{it} for each activity $i \in A$ and time period $t \in \mathcal{T}$ with the meaning that $x_{it} = 1$ if and only if the start time of i is equal to t. The start time can directly been expressed as $S_i = \sum_{i \in \mathcal{T}} t x_{i,t}$. The set $\mathcal{A}(t)$ of activities in process at a given time period t can be defined as the set of activities i that start at $\tau \in [t - p_i + 1, t]$. Hence, expression $\sum_{i \in A} \sum_{\tau=t-p_i+1}^{t} b_{ik} x_{i\tau}$ gives the amount of resource k used by all activities of $\mathcal{A}(t)$, where b_{ik} denotes the requirement of activity i on resource k (see Section 1). This allows to express the resource constraints as $|\mathcal{R}|$ knapsack inequalities per time period. This formulation is called the pulse discrete time formulation ($P-DT$) by Artigues (2017). A stronger formulation ($P-DDT$) can be obtained by reinforcing the precedence constraints, expressing them in a disaggregated way while the $P-DT$ formulation that aggregates all variables x_{it} and x_{jt} in a single constraint $S_j \geq S_i + p_i$ per precedence constraints is called the aggregated formulation. The disaggregated formulation (Christofides et al., 1987) introduces for each precedence constraint (i, j) and each time period t the implication $S_j \leq t \implies S_i \leq t - p_i$, which, given the definition of variables x_{it} is expressed by $\sum_{\tau=0}^{t} x_{j\tau} \leq \sum_{\tau=0}^{t-p_i} x_{i\tau}$. Other time-indexed formulations can be obtained by changing the semantics of the binary variable. The step discrete time formulation ($S-DT$) proposed by Pritsker and Watters (1968) and its disaggregated variant (de Souza & Wolsey, 1997; Sankaran et al., 1999; Wolsey, 1997) considers variable y_{it} where $y_{it} = 1$ iff activity i starts at t or before t. The on–off discrete time formulation ($OO-DT$) and its disaggregated variant ($OO-DDT$) introduce variable z_{it} where $z_{it} = 1$ iff activity i is in process at time t. It can be established (Sousa, 1989) that the three aggregated formulations $P-DT$, $S-DT$ and $OO-DT$ are equivalent in terms of linear relaxation as they can all be obtained from each other by linear non singular transformations. The same holds for the stronger $P-DDT$, $S-DDT$ and $OO-DDT$ formulations. Artigues (2017) reports that many formulations presented as new contributions in the literature are in fact linear transformations of either the aggregated or the disaggregated formulations, that have exactly the same relaxed solution space as the corresponding $X-DT$ or $X-DDT$ formulation, for $X \in \{P, S, OO\}$. The disaggregated time-indexed formulation is stronger than the aggregated one because if resource and integrity constraints are omitted, the resulting polytope of the disaggregated formulations is integral, which is assessed by the total unimodularity of the $S-DDT$ matrix (de Souza & Wolsey, 1997; Sankaran et al., 1999; Wolsey, 1997). Hence, the $X-DDT$ formulation are ideal formulations w.r.t. the precedence constraints. To summarize we have

$$\mathcal{P}(P-DDT) = \mathcal{P}(S-DDT) = \mathcal{P}(OO-DDT) \subsetneq \mathcal{P}(P-DT) = \mathcal{P}(S-DT) = \mathcal{P}(OO-DT)$$

Valid inequalities for time-indexed formulations The time indexed formulations can be strengthened by generating valid inequalities w.r.t. the resource constraints. Christofides et al. (1987) propose knapsack cover inequalities. A cover corresponds to a forbidden set, i.e. a set of activities C such that the sum of the activity requirements on some resource k exceeds the resource capacity. Constraint $\sum_{t \in \mathcal{T}} x_{it} \leq 1$ is valid for any activity i due to the fact that the activity can only be started once and is called a generalized upper bounding constraint (GUB). Combined with the knapsack constraint on resource k at any time t, constraint $\sum_{i \in C} \sum_{t=t-p_i+1}^{t} x_{i\tau} \leq |C| - 1$ is valid for any time period t and is called a GUB cover inequality (Cavalcante et al., 2001; Christofides et al., 1987; Sankaran et al., 1999). It simply states that at least one activity of the cover cannot be in process at time t. The inequalities are stronger if the cover is minimal, i.e. if any subset of the cover is not a cover anymore, i.e. a minimal forbidden set. Sankaran et al. (1999) propose clique inequalities that are conceptually based on a graph having a node per variable x_{it}. An edge links two nodes x_{it} and $x_{jt'}$ if scheduling activity i at time t and activity j at time t' is infeasible. This can be due to a precedence relation between i and j, to a resource constraint or to a GUB constraint if $i = j$. Then if V denotes a clique in this graph, the inequality $\sum_{(i,t) \in V} x_{i,t} \leq 1$ is a valid clique inequality. Note that in general there is an exponential number of minimal GUB cover and clique inequalities, so they must be generated as cutting planes in the branch-and-bound tree. However, given a fractional solution, finding a violated GUB cover or clique inequality is NP-hard and heuristics are used to generate the valid inequalities (Cavalcante et al., 2001; Sankaran et al., 1999). Stronger inequalities based on the same principle have been proposed more recently and strengthened by lifting procedures. Hardin et al. (2008) consider lifted cover-clique inequalities while Araujo et al. (2020) describe a large set of lifted valid inequalities as well as the way to generate them and incorporate them in a branch-and-cut method. By the time of publishing this article, optimality of 247 instances from PSPLIB was proven for the first time, which underlines the potential of MILP approaches at the expense of a quite complex algorithmic engineering.

Exponential-size time indexed formulation Another way to improve the LP relaxation of time indexed formulation by strengthening the resource constraints was proposed by Mingozzi et al. (1998) via a kind of Dantzig–Wolfe decomposition, replacing the resource constraints by a convex combination of their solutions. A feasible subset $\bar{F}$ of activities is an antichain[7] of the precedence graph such that, in addition, $\sum_{i \in \bar{F}} b_{ik} \leq B_k$ for each resource $k \in \mathcal{R}$, where B_k denotes the availability of resource k as defined in Section 1. Consider the set $\bar{\mathcal{F}}$ of all such feasible subsets. This formulation introduces binary variable ξ_{lt} if feasible subset l is in process at time t. The resource constraints can be replaced by the constraint stating that at each time t one and only one feasible subset must be executed, which is simply expressed by $\sum_{k \in \bar{\mathcal{F}}} \xi_{lt} = 1$. Then using the on–off time indexed formulation and indicator $a_{il} = 1$ if activity i is present in feasible set l, we can simply state that $z_{it} = \sum_{l \in \bar{\mathcal{F}}} a_{il} \xi_{lt}$ to obtain the formulation A-DDT with the property that $\mathcal{P}(A-DDT) \subsetneq \mathcal{P}(X-DDT)$, with $X \in \{P, S, OO\}$. Since there is an exponential number of feasible sets, the LP relaxation of this formulation is solved by column generation. This formulation was to our knowledge never used directly but its preemptive relaxation was widely exploited to obtain strong lower bounds, as detailed in Section 8.

The time indexed formulations augmented with strong valid inequalities and/or strengthened via Dantzig–Wolfe decomposition of the resource constraints yield good lower bounds and can be competitive with other exact methods to solve instances of modest size to optimality. However, due to the pseudo-polynomial number of time indexed variables of the $(X-DT)$ and $(X-DDT)$ formulations, they cannot be used for instances with a large time horizon.

[7] An antichain in a graph is a subset of nodes such that there is no path linking any pair of nodes in the subset.

Continuous time sequencing formulation based on forbidden sets Compact formulations based on a continuous representation of time have been designed to deal with this issue. The first continuous-time formulation was however of exponential size. Inspired by the disjunctive formulation of the job-shop problem Balas (1969), Alvarez-Valdes and Tamarit (1993) proposed a formulation based on continuous start-time variables S_i and ordering/sequencing variables σ_{ij} where $\sigma_{ij} = 1$ means that activity j cannot start before the end of activity i. Such a relation can be linearly expressed using the big$-M$ technique. Considering a large enough constant M_{ij}, we obtain $S_j \geq S_i + p_i - M_{ij}(1 - \sigma_{ij})$ for any activity pair (i, j). For the inequality to be valid in case $\sigma_{ij} = 0$, M_{ij} can be set to the latest completion time of i minus earliest start time of j. The precedence constraints are directly expressed by setting $\sigma_{ij} = 1$ for each precedence constraint $(i, j) \in E$. To model resource constraints, Alvarez-Valdes and Tamarit (1993) consider the set of minimal forbidden sets and state that at least a pair of activities must be sequenced inside each MFS, i.e. $\sum_{(i,j)\in \underline{F}} \sigma_{ij} \geq 1$ for each minimal forbidden set $\underline{F}$, yielding the sequencing formulation based on minimal forbidden sets ($MFS-S$). Unfortunately there is an exponential number of minimal forbidden sets, and the ($MFS-S$) is of exponential size, while the LP relaxation is really poor due to the big-M constraints.

Continuous time sequencing formulation based on resource flows A compact model can be obtained by introducing additional continuous variables representing the flow of resource units through activities. The principle, inspired again by the disjunctive graph of Balas (1969), was stated by Fortemps and Hapke (1997) but the first MILP compact formulation for the RCPSP was given by Artigues et al. (2003). Variable ϕ_{ijk} gives the number of resource units freed at the end of activity i and allocated to activity j before it starts. As flow conservation constraints, the formulation also states that the total number of entering resource units and the total number of exiting resource units of an activity i on any resource k must be equal to its requirement b_{ik}. Artificial start and end activities are considered with a zero duration and a resource requirement equal to B_k on each resource k. The link between σ_{ij} and the flow is simply written $f_{ijk} \leq \min(b_{ik}, b_{jk})\sigma_{ij}$ for each resource k. Compared to ($MFS-S$), the corresponding flow-based sequencing formulation ($\Phi-S$) is compact but its LP relaxation is even poorer.

Valid inequalities for sequencing formulations Valid inequalities were defined by Alvarez-Valdes and Tamarit (1993) and Demassey et al. (2005). A part of them are variants of the inequalities defined for the equivalent formulation in the job-shop problem by Applegate and Cook (1991) and other express lifted precedence constraints strengthened by constraint propagation. The experiments carried out by Demassey et al. (2005) reveal that the LP relaxation of the sequencing formulation remains dramatically low compared to the discrete time formulation, even if it is considerably improved by the valid inequalities.

Continuous time event-based formulations Therefore a new set of compact formulations, namely the event-based formulations, has been investigated by Koné et al. (2011). This family of formulations is inspired by the positional dates and assignment variable presented by Lasserre and Queyranne (1992), Queyranne and Schulz (1994) for the one machine problem that was extended to an event-based formulation for the flow-shop problem by Dauzère-Pérès and Lasserre (1995). A solution to the RCPSP is such that each start time of an activity is either equal to 0 or to the completion time of another activity. Hence, we have only $|\mathcal{A}|+1$ possible values for the activity start and completion times. Let us call one of these values an event and let us define continuous variable t_e for each event $e \in \{0, \ldots, |\mathcal{A}|\}$ giving the time of event e with $t_0 = 0$. The on–off event based formulation ($OO-E$) considers for each activity i and each event e, assignment variable α_{ie} such that $\alpha_{ie} = 1$ models the fact that activity i spans interval $[t_e, t_{e+1}]$. The start-end pulse event-based formulation ($P-E$) uses two binary variables β_{ie} and γ_{ie} where $\beta_{ie} = 1$ and $\gamma_{if} = 1$ mean that activity i starts at time t_e and ends at time t_f. A start-end step formulation ($S-E$) was proposed further by Tesch (2020), using two binary variables θ_{ie} and ζ_{ie} such that $\theta_{ie} = 1$ states that activity i start no later than t_e and $\zeta_{ie} = 1$ states that activity i ends no later than t_e. In the ($S - E$) formulation, the link between the event assignment and time variables is linearly expressed by $t_f \geq t_e + (\theta_{ie} + \zeta_{if} - 1)p_i$ and events are ordered A precedence constraint $(i, j) \in E$ is also simply expressed by a constraint $\theta_{je} \leq \zeta_{ie}$ at each event e. The resource constraints are established by stating that on each resource k, the sum of the requirements of activities that overlap each event e is lower than the resource capacity: $\sum_{i\in\mathcal{A}} b_{ik}(\theta_{ie} - \zeta_{ie}) \leq B_k$. Tesch (2020) improved the ($OO-E$) and the ($P-E$) formulation by new valid inequalities and also proved total unimodularity of a part of the $S-E$ formulation. He obtained that (via linear transformations) $\mathcal{P}(S-E) = \mathcal{P}(P-E) \subsetneq \mathcal{P}(OO-E)$. The ($S-E$) formulation has the "split property", meaning that only start and end events such that $f = e + 1$ need be considered for each activity. From this property, an equivalence of the LP relaxation of ($S-E$) with a linear program presented by Carlier and Néron (2003) is established (see Section 8). He also introduced the interval start end event-based formulation ($I-E$) using variable ω_{ief} equal to 1 if activity i starts at event e and ends at event f. He showed the new formulation dominates the other event-based formulations since via a linear transformation $\mathcal{P}(I-E) \subsetneq \mathcal{P}(S-E)$. The relation between ($I-E$) and ($P-DDT$) is also studied with no-dominance between both formulations. It is shown that performing an expansion and then a restriction of the number of events in ($I-E$) allows to obtain formulation (DDT). In terms of computational experiments, the best event-based formulation is ($S-E$), which obtains better optimality gaps than (DDT) on the 60-activity instances from the PSPLIB. A promising perspectives would be to find strong valid inequalities for the event-based formulations and then to apply powerful branch-and-cut algorithm engineering (Araujo et al., 2020).

5. Constraint programming, SAT

In constraint programming (CP), a problem declaration is decomposed into constraints on finite-domain decision variables. Each constraint is associated with a dedicated filtering algorithm, which uses consistency tests allowing to reduce the domain of the decision variables involved in the constraint. CP is characterized by a rich constraint language, some constraints being elementary such as a linear constraint on n variables, some other, called global constraints (van Hoeve & Katriel, 2006), capturing a complex relation between a non-fixed number of variables. Once the problem is declared, the principle is to solve the so-described constraint satisfaction problem (CSP) by a tree search algorithm in which branching assigns values to decision variables, such as in a branch and bound tree. At each node, the filtering algorithms of the different constraints are orchestrated, which is often named the constraint propagation process, to reduce the domains of the decision variables taking account of the decisions made so far, and possibly detecting an global inconsistency that allows to prune the node. For more details on CP, we refer to Rossi et al. (2006) and for general descriptions of applications of CP to scheduling, we refer to Dorndorf et al. (2000b) and Baptiste et al. (2001).

For the RCPSP, the standard CP formulation uses the start time decision variables S_i, $i \in \mathcal{A}$ whose initial domain is the interval $[0, T - p_i]$ and two sets of constraints: the first set include the binary precedence constraints $S_j \geq S_i + p_i$, $\forall(i, j) \in E$ and the second set defines a global constraint for each resource denoted as Cumulative. This global constraint was defined by Aggoun and Beldiceanu (1993). The global constraint ensures that the capacity of each resource $k \in \mathcal{R}$ is not exceeded at each time point belonging to the domain of at least one task, which is expressed by Cumulative$(S, p, b_{.k}, B_k)$, where S, p, and $b_{.k}$ are the start time, duration and requirement vectors, respectively, while B_k is the capacity of resource k. Declaring a cumulative constraint for each resource to a CP solver is equivalent to enforcing the resource constraints of the RCPSP.

The considered CSP aims at finding a schedule of makespan of at most T. Optimization can be performed by a binary search on T. In the search tree, branching either assigns values to the start times or make decisions on the relative ordering of activities, such as in the branch-and-bound approaches presented in Section 6. At each node, constraint propagation iteratively runs the filtering/consistency checking algorithms of the precedence and cumulative constraints. In CP theory, different levels of consistencies have been defined, such as the generalized arc consistency that ensures that for any time t in the domain of a start time variable S_i, there exists a support, i.e. an assignment of all other start times to some values in their domains that satisfies each constraint separately. This level of inconsistency generally is too costly and in most CP-based scheduling implementations, the domain of a start time is represented by an interval $[ES_i, LS_i]$ where ES_i is the earliest start time of activity i and LS_i is its latest start time given the makespan T. In this case only bound consistency, i.e. the existence for a support for the bounds ES_i and LS_i, is enforced. Note that the earliest completion time $EC_i = ES_i + p_i$ and the latest completion time $LC_i = LS_i + p_i$ are also values of interest. The principle of applying constraint programming and SAT approaches to the RCPSP has been described by Baptiste et al. (2001), Brucker and Knust (2012), Dorndorf (2002), Hebrard (2017), Schutt et al. (2015). In Section 5.1 the precedence and resource constraint propagation and the associated tradeoffs are introduced. Section 5.2 presents the main resource-constraints consistency checks for the cumulative constraint while Section 5.3 presents other variants of consistency checks, based on relaxations, special cases and redundant constraints. Finally, Section 5.4 presents how the consistency tests are included in the CP search process and in hybrid CP/SAT approaches.

5.1. Constraint propagation and consistency checks trade-off: inference vs speed

For precedence constraints, bound consistency can be achieved by different polynomial algorithms that ensure that ES_i is not lower than length of the longest path entering i in the precedence graph and LS_i is not larger than T minus the length of the longest path issued from i in the precedence graph (Dorndorf, 2002).

Unfortunately, ensuring bound consistency for the cumulative constraint is NP-hard (Baptiste et al., 2001). Several filtering algorithms achieving lower levels of consistency were proposed for the cumulative constraint. The principle of using sets of activities in conflict on a cumulative resource to necessary precedence constraints and update activity time windows was stated in the pioneering work of Erschler (1976), Erschler et al. (1979). An important concept is the compulsory part of an activity inside a given interval, i.e. given the activity time window, the minimum overlapping duration of the activity with the interval. The concept was defined by Lahrichi (1982) and the first consistency checking and time window adjustment rules were proposed by Lopez (1991) under the term "energetic reasoning". The fundamental consistency check is the energetic overload check that can be defined given a time interval $[t_1, t_2]$. The minimum overlap length of the execution interval of an activity i with $[t_1, t_2]$ is either equal to $t_2 - t_1$ if $[t_1, t_2] \subseteq [LS_i, EC_i]$, p_i if $[ES_i, LC_i] \subseteq [t_1, t_2]$, $EC_i - t_1$ if the minimum overlap is reached when left-shifting the activity and $t_2 - LS_i$ if the minimum overlap is reached when right-shifting the activity. The minimum of these 4 values when they are all positive gives the compulsory part of activity i. By multiplying the compulsory part of an activity i in $[t_1, t_2]$ by its requirement on some resource k, we have the maximum required "energy" consumption e_{i,k,t_1,t_2} of the activity inside the interval. The maximum required energy of set Ω is $e_{\Omega,k,t_1,t_2} = \sum_{i\in\Omega} e_{i,k,t_1,t_2}$. Then, the energy slack of resource k for activity set Ω in interval $[t_1, t_2]$ is $s(\Omega, k, t_1, t_2) = (t_2 - t_1)B_k - e_{\Omega,t,t_1,t_2}$. If the slack is negative then no solution exists.

This fundamental property gave rise to a wide variety of consistency checking algorithms, regrouped under the name of interval consistency tests by Dorndorf (2002). These tests differ by the type of considered $[t_1, t_2]$ intervals and Ω sets. Generally, each type of consistency test comes with an overload check, an earliest start adjustment rule and a symmetric latest completion adjustment rule. The consistency checking algorithms of interest offer a trade-off between inference power and computational complexity. In the next subsection, the only main consistency tests for resources constraints are presented and only the earliest time adjustments are presented.

5.2. Consistency tests for the cumulative constraint

Timetable constraint propagation The timetable consistency test (Caseau & Laburthe, 1996; Le Pape, 1994, 1995; Nuijten, 1994) considers intervals of unit length $[t, t+1]$ and is thus called unit-interval consistency test by Dorndorf (2002). In this case $e_{i,k,t,t+1} = b_{ik}$ if $LS_i \leq t < EC_i$ and 0 otherwise. The overload check consists in verifying whether $s(\mathcal{A}, k, t, t+1) \geq 0$ for all resource k and time t. For the time bound adjustment, given an activity i, if $s(\mathcal{A} \setminus i, k, t, t+1) < b_{ik}$ for some time $t < \min(LS_i, EC_i)$ and resource k, then the earliest start time of activity i cannot be lower than t. Focusing on the time bounds of each activity, the overall test can be performed in $O(|\mathcal{A}|^2)$. Efficient sweep techniques to achieved the filtering in a better average complexity is proposed by Letort et al. (2012). Ouellet and Quimper (2013) reduce this complexity to $O(|\mathcal{A}| \log |\mathcal{A}|)$ using an AVL-tree to store relevant intervals while Fahimi and Quimper (2014) obtain an $O(|\mathcal{A}|)$ using a union find data structure. Gay et al. (2015) propose an $O(|\mathcal{A}|^2)$ algorithm whose advantage is its simplicity and scalability.

Reasoning on task intervals Another family of interval consistency tests, the cumulative overload check, the cumulative edge-finding and not-first/not-last adjustments were introduced by Nuijten (1994) inspired by their counterpart for disjunctive resources. The adjustments reason on the position of an activity i relatively to a set of activities Ω not including i on a resource k. The considered intervals are called "task intervals" by Caseau and Laburthe (1996). A task interval relative to a set Ω on a resource k is the smallest interval $[ES_\Omega, LC_\Omega]$ such that each task $j \in \Omega$ has a maximal energy consumption, i.e. $p_j b_{jk}$, in the interval. As the interval depends on Ω, we will denote $e_{\Omega,k,ES_\Omega,LC_\Omega}$ as $e_{\Omega k}$. Given the current time bounds, the energy is maximal if $ES_\Omega = \min_{i\in\Omega} ES_i$ and $LC_\Omega = \max_{i\in\Omega} LC_i$ and $e_{\Omega k} = \sum_{i\in\Omega} e_{ik}$ with $e_{ik} = p_i \cdot b_{ik}$. A relaxation of the task interval concept was introduced by Vilım (2007) as the left cut of an activity i relatively to a set Ω and was proved useful in overload check and edge finding algorithms. Namely, $Lcut(\Omega, i)$ is the set of activities j in Ω such that $LC_j \leq LC_i$.

Cumulative overload check The overload check consists in evaluating the sign of the slack $s(\omega, k, ES_\omega, LC_\omega)$, i.e. testing if $e_{\omega k} \leq B_k(LC_\omega - ES_\omega)$ for all subsets $\omega \subseteq \mathcal{A}$ on a resource k. If the slack is not negative, the instance is said to be E-feasible. Interestingly, this is equivalent to finding a solution to the fully elastic relaxation where each task requirement can vary between 0 and B_k provided that it occupies energy e_{ik} inside its time window. A link between the fully elastic relaxation and the preemptive one machine problem has been established by Baptiste et al. (2001). A one-machine instance can be obtained by defining an activity i' for each activity i with release date $r_{i'} = B_k ES_i$, deadline $\tilde{d}_{i'} = B_k LC_i$ and duration $p_{i'} = e_{ik}$. Consequently the existence of a solution can be checked in $O(|\mathcal{A}| \log |\mathcal{A}|)$ by the Jackson's preemptive scheduling algorithm. Note that value $B_k ES_i + e_{ik}$, i.e. the earliest completion time of i in the one machine instance, corresponds to the maximal consumption of the resource up to the completion of i in a fully elastic schedule. It was defined as the energy envelope $Env(i, k)$ of an activity by Vilím (2009b) who also defined the energy envelope of a set of activities Ω as $Env(\Omega, k) = max_{\omega\subseteq\Omega}(B_k ES_\omega + e_{\omega k})$. He remarked that the overload check can be rewritten for each activity $i \in \mathcal{A}$, $Env(Lcut(\mathcal{A}, i), k) \leq B_k LC_i$. Thanks to the so-called Θ-tree structure, a balanced binary tree where activities are sorted in non-decreasing order of ES_i, the overload check can also be performed

in $O(|\mathcal{A}|\log|\mathcal{A}|)$ as for the single machine case (Vilím, 2004; Wolf & Schrader, 2005). Fahimi and Quimper (2014) replaced the Θ-tree data structure by the "timeline" data structure based on union find structure that allow many operations to be made in constant time and obtain a linear complexity. A stronger overload check was proposed by Gingras and Quimper (2016) based on rewriting it as $Ec_\omega \leq LC_\omega$ where Ec_ω is the earliest completion time of activity set ω. Since the exact value of Ec_ω is NP-hard to compute, the "standard" overload check sets this value to $ES_\omega + \lceil e_\Omega / Bk \rceil$ or equivalently with the envelope notation to $\lceil Env(\Omega)/B_k \rceil$, which corresponds to the fully elastic relaxation of the activities. Gingras and Quimper (2016) define a stronger "horizontally elastic" relaxation where an activity may require between 0 and b_{ik} units at each time point. Using a clever "profile" data structure, Gingras and Quimper (2016) obtain a stronger overload check in $O(|\mathcal{A}|^2)$.

Cumulative edge finding Concerning the edge finding rule, it can be simply stated as asking, given a set of activities Ω and an activity i, whether the lower bound on the earliest completion time $EC_{\Omega\cup i} = ES_{\Omega\cup i} + e_{\Omega\cup i}/B_k$ exceeds the latest completion LC_Ω. If this is the case, then activity i must end strictly after all activities of Ω and the earliest date of i can be increased. For any subset $\omega \subseteq \Omega$, in interval $[ES_\omega, LC_\omega]$, the available energy for the activities of ω allowing any start time of i is at best $\rho(\omega, i) = (LC_\omega - ES_\omega)(B_k - b_{ik})$. Since activities in ω require an energy of $e_{\omega k}$, a lower bound of the earliest start time of i is $\max_{\omega\subseteq\Omega} ES_\omega + \lceil \max\{0, (e_{\omega k} - \rho(\omega, i)/b_{ik})\} \rceil$. An $O(|\mathcal{A}|^2)$ algorithm was given by Baptiste et al. (2001), while Mercier and Van Hentenryck (2008) proved that the previous algorithm is incomplete and provided a complete variant in $O(\kappa|\mathcal{A}|^2)$, where $\kappa \leq |\mathcal{A}|$ is the number of distinct capacity requirements of the tasks. This algorithm was further improved with an even lower complexity of $O(\kappa|\mathcal{A}|\log|\mathcal{A}|)$ (Vilím, 2009a) using the energy envelope expression and the Θ-trees. An $O(|\mathcal{A}|^2)$ variant was proposed by Kameugne et al. (2014) based on minimum slack and maximum density concepts. It has to be noted that all these algorithms are not idempotent, i.e. they may need several iterations to converge to the same fixed point, and the latter algorithm may need more iterations than the one of Vilím (2009a). Under the same principle as with the overload check, Gingras and Quimper (2016) derive a stronger edge finder using the horizontally elastic relaxation in $O(\kappa|\mathcal{A}|^2)$. Kameugne et al. (2023) extend the minimum slack and maximum density approach of Kameugne et al. (2014) to obtain an $O(\mathcal{A}^2)$ competitive "slack-density horizontally elastic" edge finder.

Cumulative extended edge finding The extended edge-finding rule (Nuijten, 1994) consider the case where an activity i starts at its earliest start time and overlaps task interval $[ES_\Omega, LC_\Omega]$ such that there is not enough energy available in the interval ($ES_i \leq ES_\Omega \leq EC_i$ and $e_{\Omega k} + b_{ik}(EC_i - ES_\Omega) > B_k(LC_\Omega - ES_\Omega)$). In this case, i must end after all activities in Ω, and the above-described ES_i adjustment can be applied. Mercier and Van Hentenryck (2008) propose a $O(\kappa|\mathcal{A}|^2)$ algorithm. Ouellet and Quimper (2013) propose an extension of the edge finder of Vilím (2009a) based on new filtering rules allowing to reach a complexity of $O(\kappa|\mathcal{A}|\log|\mathcal{A}|)$.

Cumulative not first/not last The not first rule gives conditions such that an activity i has to be scheduled after at least one activity of a set Ω, which allows increasing ES_i. If $ES_\Omega \leq ES_i \leq \min_{j\in\Omega} EC_j$ and if $e_{\Omega,k} + b_{ik}(\min(EC_i, LC_\Omega) - ES_\Omega) > B_k(LC_\Omega - ES_\Omega)$ then $\min_{j\in\Omega} EC_j$ is a valid lower bound for ES_i. Schutt et al. (2005) found a mistake in the original algorithm by Nuijten (1994) and proposed a complete filtering algorithm in $O(|\mathcal{A}|^3\log|\mathcal{A}|)$ based on Θ-trees. Kameugne and Fotso (2010) reduce this complexity to $O(|\mathcal{A}|^2 H\log|\mathcal{A}|)$ where H is the maximum between the number of distinct earliest completion and latest start times of tasks, while Kameugne and Fotso (2013) obtain the same fixed point with possibly more iterations with an $O(|\mathcal{A}|^2\log|\mathcal{A}|)$ algorithm. Fahimi et al. (2018) further reduce the complexity of the inner incomplete (non idempotent) algorithm by a $\log|\mathcal{A}|$ factor thanks to a union-find data structure, yielding a quadratic complexity. Recently, Kameugne et al. (2018) integrate the horizontally elastic relaxation into the not first/not last test yielding a stronger algorithm running in $O(|\mathcal{A}|^3)$.

Energetic reasoning overload check The last family of interval consistency tests consists in the energetic reasoning test originally presented by Lopez (1991). This test is called the left shift/right shift propagation technique by Baptiste et al. (1999) and include the energetic overload check as presented above and the energetic time adjustment rules. As stated above, the energetic overload check consists in testing if for all intervals $[t_1, t_2]$, and for any resource k, the energy slack $s(\mathcal{A}, k, t_1, t_2)$ remains positive. Baptiste et al. (1999) show that the slack function is a continuous and piecewise linear function of t_1 and t_2 whose minimum is necessarily reached in a breakpoint. They bound the number of breakpoints by a quadratic number of relevant intervals. An $O(|\mathcal{A}|^2)$ algorithm is proposed. Based on a study by Schwindt (1998), Derrien and Petit (2014) further study the local minima of the slack function and manage to reduce the number of relevant intervals by a factor of 7 without decreasing the worst-case time complexity. Ouellet and Quimper (2018) show that for a resource k and a given interval $[t_1, t_2]$, the required energy $e(\mathcal{A}, k, t_1, t_2)$ can be computed in $\log|\mathcal{A}|$ time after an $|\mathcal{A}|\log|\mathcal{A}|$ preprocessing step using the range tree data structure. Furthermore, based on the fact that $e(\mathcal{A}, k, t_1, t_2)$ is a Monge matrix and the slack $s(\mathcal{A}, k, t_1, t_2)$ is a reverse Monge matrix, the authors prove that only $O(|\mathcal{A}|\log|\mathcal{A}|)$ intervals need be checking, reducing the overload check complexity to $O(|\mathcal{A}|\log^2|\mathcal{A}|)$. These properties are further exploited by Carlier et al. (2022) to obtain an $O(|\mathcal{A}|\alpha(|\mathcal{A}|)\log|\mathcal{A}|)$ algorithm, where $\alpha(.)$ is Ackermann's inverse function.

Energetic reasoning adjustments The energetic filtering rule (Lopez, 1991) evaluates the feasibility of starting an activity at its earliest start time w.r.t. an interval $[t_1, t_2]$ and a resource k. By doing so, activity i occupies an energy $l(i, t_1, t_2) = \max(0, \min(t_2 - t_1, EC_i - t_1))b_{ik}$ in the interval. If this quantity is larger than the slack $s(\mathcal{A}\setminus i, k, t_1, t_2)$ then a valid lower bound on the earliest start time of i is $\lceil t_2 - s(\mathcal{A}\setminus i, k, t_1, t_2)/b_{ik} \rceil$. Baptiste et al. (1999) proposed an $O(|\mathcal{A}|^3)$ filtering algorithm. The reduction of the number intervals of interests exhibited by Derrien and Petit (2014) improved the experimental running times. The geometrical interpretation of energetic reasoning allows Bonifas (2017) to compute the maximum possible adjustment for at least one activity in $O(|\mathcal{A}|^2\log|\mathcal{A}|)$, which was improved by Tesch (2016) to obtain a possible adjustment on all activities with the same complexity and by Carlier et al. (2020) to obtain the same adjustments in $O(|\mathcal{A}|^2)$. These algorithms must be ran several times to reach the same fix point than the algorithm of Baptiste et al. (1999). A complete energetic reasoning filtering in $O(|\mathcal{A}|^2\log^2|\mathcal{A}|)$ was obtained by Ouellet and Quimper (2018) and in $O(|\mathcal{A}|^2\log^2|\mathcal{A}|)$ by Tesch (2018).

Dominance relations between consistency tests and compound rules In Baptiste et al. (2001), dominance relations are stated in the sense that a consistency test C dominates another one C' if all deductions made by C' are also made by C. The timetabling, edge-finding, extended edge finding and not/first not/last consistency checks are not comparable while energetic reasoning dominates all other tests except the not first/not last one. The integration of non dominated rules inside the same filtering algorithm while keeping a low time complexity can pay off. The timetable, the edge finding and the extended edge finding rules were integrated by Vilím (2011) who proposed the (non idempotent) timetable (extended) edge finding algorithm running in $\mathcal{O}(|\mathcal{A}|^2)$ time. Ouellet and Quimper (2013) obtained an algorithm running in $O(\kappa|\mathcal{A}|\log|\mathcal{A}|)$ time and performing more adjustments at each iteration. Tesch (2018) proposed an $O(|\mathcal{A}|^2\log|\mathcal{A}|)$ algorithm that integrated energetic reasoning and edge finding, with a relaxed $O(|\mathcal{A}|^2)$ variant that subsumes edge-finding while also performing incomplete energetic adjustments. Kameugne et al. (2013) propose an extended edge finding algorithm that also integrate such incomplete energetic adjustments running in $O(|\mathcal{A}|^3)$. All of the above-presented consistency checks focus solely on the cumulative constraint part of the problem. However taking account of the interaction of precedence and resource constraints is a key issue for efficient RCPSP solving. Laborie (2003) proposed the energy-precedence constraint that states that given an

activity i a resource k and a set Ω of predecessors, i cannot start before $ES_\Omega + \lceil e_{\Omega,k}/B_k \rceil$. An $O(|\mathcal{A}|(\lambda + \log|\mathcal{A}|))$ filtering algorithm is proposed where λ is the maximal number of predecessors of an activity.

5.3. More consistency tests

Applying consistency tests from relaxations Other consistency tests can be applied for the cumulative constraints even if they were conceived for relaxations or special cases. Obviously, any consistency test valid for a relaxation of the decisional RCPSP is also valid for the decisional RCPSP. Baptiste et al. (1999), consider three relaxations of the cumulative constraint: fully elastic, partially elastic and preemptive. Variants of the timetable, (extended) edge finding and energetic reasoning consistency tests exist for the relaxed versions and are weaker than their counterpart for the non preemptive cumulative constraints. However they may offer a good compromise in terms of computational effort. Carlier and Pinson (2004) suggest the use of the Jackson's pseudo-preemptive scheduling algorithm to perform adjustments for the cumulative constraint, based on the pseudo-preemptive relaxation where several activities can share the same resource unit yielding non necessarily integer resource requirements.

Applying consistency tests from special cases Applying consistency tests for a special cases is also possible if this special case is detected, which may happen after decisions were taken along the search tree. Notably, a very efficient set of consistency sets have been proposed for the disjunctive constraint (i.e. a cumulative constraint with $B_k = 1$). Whenever two activities i and j are incompatible w.r.t. a resource, i.e. when $b_{ik} + b_{jk} > B_k$ then disjunctive constraint propagation can be applied to activity pair (i, j). Actually, the disjunctive constraint propagation algorithm, such as disjunctive edge finding initiated in Carlier and Pinson (1989) for the jobshop scheduling problem, are very powerful when large sets of activities are in disjunction. Hence, several authors (Baptiste & Pape, 2000; Brucker et al., 1998a; Dorndorf et al., 2000a) propose to search for maximal cliques in an incompatibility graph where nodes represent the activities and an edge represents either a resource incompatibility or a precedence relation, each maximal clique yielding a disjunctive set on which the disjunctive consistency checks can be applied. Brucker et al. (1998a) proposed the symmetric triple rule where a symmetric triple is a forbidden set $\{i, j, k\}$ where i must overlap with k and j must overlap with k. Then i and j are in disjunction and other deductions can be made by considering the relative positioning of a fourth activity w.r.t. a symmetric triple.

Generating redundant cumulative constraints Generating redundant constraints is of interest in constraint programming in the case where the redundant constraint to be added has a potentially stronger filtering than the already present constraints. The above-presented generation of implied disjunctive constraint is a typical example. Redundant cumulative constraints can be generated with the objective to obtain tighter constraints in terms of lower resource capacities or larger resource requirements, or both, in order to obtain stronger adjustments. Baptiste and Bonifas (2018) present a way to obtain redundant cumulative constraints based on dual solutions of a linear program that formulates the preemptive cumulative problem. Their method yield stronger constraints than the method proposed in Carlier and Néron (2003, 2007) based on dual feasible functions.

5.4. Constraint programming and boolean satisfiability search methods and solvers

Some constraint programming approaches rely on problem-agnostic branching heuristics such as DomOverWDeg (Boussemart et al., 2004). This is most often the case when the goal is to evaluate the merits of new consistency checks or more efficient implementations of standard ones, such as in Ouellet and Quimper (2013) rather than competing with the state-of-the-art dedicated branch-and-bound methods presented in Section 6. Other CP search approaches use depth-first chronological backtracking. In particular there are similarities between (Baptiste & Pape, 2000; Caseau & Laburthe, 1996; Dorndorf et al., 2000a; Le Pape et al., 1994) who all generate a binary search tree. On the left branch an activity i with a minimum earliest start time is selected to be scheduled at ES_i. Upon failure, on the right branch the activity earliest start time is either increased (Baptiste & Pape, 2000; Caseau & Laburthe, 1996; Dorndorf et al., 2000a; Liess & Michelon, 2008) or marked non selectable until its earliest start time has increased (Le Pape et al., 1994). Baptiste and Pape (2000) propose to set ES_i to the earliest completion time of the set of activities such that $ES_j < EC_i$. Non chronological branching schemes have also been experimented in CP approaches. In Brucker et al. (1998a), a branching scheme is based on the property that if for any pair of activities if it is known whether the activities cannot overlap or must overlap then a feasible schedule can be obtained in polynomial time if it exists. Then at each node, a pair of activities whose overlapping status is unknown is selected and two child nodes are created, one where the activities must overlap and the other one where the activities cannot overlap. In Laborie (2005), branching consists in selecting a minimal forbidden set and generating a child node for each precedence constraints that destroys the MFS. All these branching schemes have the objective to boost the above-presented consistency checks. Except for the last approach, none of the exact CP-based solvers were able to compete with the dedicated branch-and-bound approach by Demeulemeester and Herroelen (1997b) on the PSPLIB instances (Kolisch & Sprecher, 1997). This is partly due to the special nature of these instance set, on which the hardest instances are highly disjunctive (Baptiste & Pape, 2000). But it must also been remarked that the consistency checks for the cumulative constraint are not as efficient as their disjunctive counterpart that make standard CP as the technique of choice for the jobshop problem and its variants.

For these problems, the importance of guiding the search by conflict analyses has been underlined (Grimes & Hebrard, 2015). This was also suggested for the RCPSP by the efficiency of the cutset dominance rule by Demeulemeester and Herroelen (1997b) based on specific "no-goods" learning. A SAT approach coupled with on the fly generation of cover clauses to deal with resource constraints was proposed by Horbach (2010), based on time indexed encodings, i.e. using literal u_{it} equivalent to the MIP on–off z_{it} variable and literal s_{it} equivalent to the MIP start pulse variable x_{it} (see Section 4). Motivated by the success of boolean satisfiability (SAT) solvers, lazy clause generation (Feydy & Stuckey, 2009; Ohrimenko et al., 2009) is an hybrid method in which a CP solver maintains a clause data base issued from the explanations of the constraint propagation. Literals that encode the integer decision variables involved in the explanation and the related clauses are generated on-the-fly (lazily) and are transmitted to a SAT solver embedded in the CP solver as a constraint propagator. The SAT engine performs unit propagation and conflict analysis and update the clause base, which in turn transmits domain reductions to the CP solver variables. Schutt et al. (2009, 2011) applied this technique to solve the RCPSP using explanations of the timetable and (extended) edge finding propagators. In Schutt et al. (2013), the same technique is applied with explanations of the time table extended edge finding. Two SAT encodings are proposed: a time indexed encoding similar to the one of Horbach (2010) but using also step variables equivalent to the MIP y_{it} variables, and also a task-indexed encoding using a boolean variable l_{ij} meaning that when i starts, j is in process. The challenge for explaining a propagator is to have the smallest clauses possible to increase their generality. A related approach lies in satisfiability modulo theory solvers which are basically extensions of SAT solvers allowing other constraints than standard clauses, e.g. constraints involving integer variables, that are evaluated via a so-called "background theory". Such approaches were proposed in Ansótegui et al. (2011), Bofill et al. (2020). The theory used in the latter paper is the integer difference logic theory to encode

the precedence constraints while SAT encodings are used for resource constraints. The above-presented CP/SAT approach obtain state-of-the-art results on various problem instances often outperforming in terms of exact solving the standard CP approaches and the former dedicated branch and bound approaches. To go further, a promising research area is certainly to integrate mixed-integer linear components into the CP/SAT solvers.

Constraint programming solvers Some of the above-defined constraint programming approaches are integrated into academic or commercial solvers. As already mentioned, the CHIP solver was the first to address the RCPSP via the cumulative constraint (Aggoun & Beldiceanu, 1993; Beldiceanu et al., 1996). Various publications present the constraint programming elements embedded into the ILOG Schedule solver (Le Pape et al., 1994) that became ILOG Scheduler (Laborie, 2003) and finally IBM CPoptimizer (Vilím et al., 2015). The Chuffed (Chu et al., 2018) and Google OR-Tools CP-SAT (Perron & Didier, 2023) solvers implement lazy clause generation. SCIP (Berthold et al., 2010) is an hybrid solver implementing CP/SAT an MIP. More recently, Hexaly (formerly LocalSolver) is not considered as a constraint programming solver but integrates disjunctive and cumulative constraint propagation algorithms (Blaise et al., 2020; Laborie, 2023) allowing to solve the RCPSP. Other constraint programming solvers such as OptalCP (Heinz et al., 2024) and choco-solver (Prud'homme & Fages, 2022) report result on the RCPSP.

6. Branch-and-bound methods

At the end of the last century, branch-and-bound procedures were much more investigated than today. That may be because meta-heuristics were much less known than they are today, and they also were not yet powerful enough to generate very good solutions (although the times changed quickly, as will be discussed in Section 7). Moreover, at that time there were not many and powerful generic solvers available, so researchers were forced to develop the often very specific and fine-tuned components of a branch-and-bound procedure. Although branch-and-bound procedures often lead to very good, and theoretically optimal, solutions, the challenge of such procedures is to fine-tune the various algorithmic components with the characteristics of the problem. It is therefore often difficult to generalize these procedures so that they can easily deal with a slightly modified variant of the notorious scheduling problem. After the development and publication of some very good specific branch-and-bound procedures, which remain highly competitive to this day, the attention waned and most researchers began either to focus their research time on metaheuristics to push the RCPSP to near-optimality (cf. Section 7) or on more generic CP/SAT/MILP approaches (see Sections 4 and 5).

Despite the declining interest of many researchers in developing specific exact methods to solve the RCPSP, the focus of branch-and-bound formulations is far from having disappeared, especially since this exact search method is exploited by the above-mentioned CP, SAT and MILP solvers. A few years ago Morrison et al. (2016) drew our attention to the fact that BnB procedures still have many challenges. They provided a survey of recent advances in searching, branching, and pruning for branch-and-bound algorithms, without particularly focusing on the project scheduling literature. At the end of their study, the authors gave some guidelines for future research that might be interesting and relevant for the RCPSP research. More recently, Coelho and Vanhoucke (2018) have implemented some of these ideas and proposed a so-called *exact composite lower bound strategy* that brought together the existing components of the most competitive BnB procedures to solve the RCPSP. These components can be divided into the following four categories:

- Search strategy: The strategy to explore the tree can be done using a depth-first strategy (which is mostly used in the literature) but also by best-first, breadth-first or a hybrid strategies. Moreover, each strategy can be implemented as the upper bound strategy (U) or the minimum lower bound strategy (L), as proposed by Patterson (1974).
- Branching scheme: The branching scheme determines how nodes are constructed at each level of the tree, with three main strategies: activity start time branching (A), serial branching (S), and parallel branching (P). The activity start time branching strategy involves selecting one activity and branching based on its possible start times within its allowable time window. For each node, the algorithm creates branches corresponding to the different feasible start times of the chosen activity, thus exploring different scheduling scenarios for that activity. In the serial branching scheme, activities are selected one by one at their earliest possible start times. The algorithm follows an activity incrementation approach, where it iteratively schedules eligible activities. This means that it selects an activity, schedules it, and then moves on to the next activity, continuing this process until all activities are scheduled. The parallel branching approach uses a time incrementation method. The algorithm iteratively increases the time pointer in the schedule, looking for activities that can be scheduled at each time increment. Unlike serial branching, which focuses on scheduling activities one by one, parallel branching advances through time and at each time step considers all activities that can start at that particular time.
- Branching order: Every time a node is selected for branching, a number of child nodes are constructed and a ranking based is determined to select which child node is selected first (and next) for further branching. This selection can be done using the best lower bound (B), the activity ID (A), the minimum time window (M) or completely random (R).
- Lower bounds: The lower bounds are proposed by Klein and Scholl (1999) and combined into a composite version classified in 4 categories. The version CLB0 only calculates the critical path lower bound at each node and is standard in most BnB procedures. The other versions add 4, 8 or 12 other lower bounds on top of this fast and efficient critical path lower bound, and only a few BnB procedures in the literature make use of more than one lower bound.

Since this study builds on the existing components of the literature, it is tempting to think that such a study could not provide many innovative insights compared to the existing work. However, by combining all components, the authors ended up with 48 different configurations. The depth-first strategy was always chosen because it generally delivers much better performance. A selection was made for the other components, as follows:

[U,L] ×[A,P,S] ×[B,A] ×[0,4,8,12]

which eventually led to 48 combinations, i.e. 48 different BnB configurations.

Besides the implementation of these 48 configurations, the authors also extended the BnB procedure with a so-called *credit/tabu list system* to dynamically add and remove lower bounds in the branch-and-bound procedure. Although stronger lower bounds can lead to improvements in the search, they also require more CPU time, and the new credit/tabu list system should therefore allow better fine-tuning of this quality/computation trade-off. After a thorough further analysis of the existing BnB procedures in Guo et al. (2023) based on 12 existing procedures from the literature, it could be deduced that only 9 of these 48 configurations were investigated in the literature as presented in Table 1. Not every branch-and-bound procedure fits perfectly in the CLB framework because of the minor differences with the specific CLB implementation, and the procedures of Brucker et al. (1998b) and Dorndorf et al. (2000a) contain additional elements (not indicated in the table) that are too different to claim they are fully included in the CLB framework.

Above all, the study has shown what was already mentioned by Morrison et al. (2016), namely that there is still plenty of room for

Table 1
The components of branch-and-bound procedures in the literature.

References	Search strategy (1)				Search strategy (2)			Branching scheme			Branching order				Lower bounds		
	Depth	Best	Breadth	Hybrid	U	L	D	A	P	S	B	M	R	A	cp	cp+1	cp+12
Patterson (1974)	×				×	×	×	×			–	–	–	–	×		
Stinson et al. (1978)		×			×				×		×					×	
Talbot and Patterson (1978)	×				×			×						×	×		
Christofides et al. (1987)	×				×				×					×		×	
Bell and Park (1990)		×			×				×		×				×		
Demeulemeester and Herroelen (1992)	×				×				×		×					×	
Mingozzi et al. (1998)	×				×				×					×		×	
Demeulemeester and Herroelen (1997a)	×	×		×	×				×		×					×	
Brucker et al. (1998b)	×				×						×					×	
Nazareth et al. (1999)		×	×		×				×					×	×		
Dorndorf et al. (2000a)	×				×			×				×			×		
Sprecher (2000)	×				×					×				×		×	
Coelho and Vanhoucke (2018)	×				×	×	×	×	×	×	×	×	×	×			×

improvement, even for a problem like the RCPSP, which has been the subject of many decades of research. More specifically, in a computational experiment on the PSPLIB instances, it was demonstrated that the parallel branching scheme (PAR) could optimally solve more instances within a 1-hour time limit compared to the other branching schemes (SER, AST), as already known in the literature. However, the other branching schemes were not completely dominated by this well-performing branching scheme, as both the SER and AST branching schemes could find solutions that the PAR branching scheme could not. These results indicate that a carefully elaborated combination of different strategies has the potential for improvements, and that bringing together existing academic research can lead to new enhancements and insights.

Furthermore, the experiments also showed that there is still much room to find further improvements. Specifically, the experiments were repeated using the CSS procedure as a combined search strategy, alternating between the different branching schemes (dividing the total 1-hour time limit among the three branching schemes for 20 min each). Despite the fact that this CSS procedure could find nearly all solutions found by the PAR, SER, and AST procedures together, there were still 29% of the PSPLIB instances for which no optimal solutions could be found, demonstrating that this challenging scheduling problem still holds many secrets. Since the experiment had shown that a combination of different branching schemes can lead to better solutions, Guo et al. (2023) proposed a method for ranking and automatically selecting the best performing configuration(s) based on network and resource characteristics of the problem instance. They utilized two regression methods (kNN and ridge regression). Their findings consistently demonstrated that these machine learning methods outperform any single best configuration and often approach the optimal solution achievable by running each configuration separately, albeit at a significantly higher computational cost and running time.

Finally, with the recent attention to *matheuristics*, the importance of exact methods, such as the branch-and-bound methods discussed in this section, may increase further. Since matheuristics by definition try to combine the best elements of exact and heuristic methods (see Section 7), a correct and good understanding of both methods will be necessary to make these matheuristics performable. It may be a matter of time before the MIP and BnB procedures will access matheuristics to solve the RCPSP, hopefully with enhancements that will not only provide better solutions, but also fresh insights into the ongoing search for exact solution procedures for this challenging scheduling problem.

7. Heuristics and metaheuristics

Given the computational complexity of the RCPSP, the development of heuristics has always been a natural research direction. The first approaches were priority rule-based heuristics, which can be traced back to the 1960s. The development of metaheuristics for the RCPSP started in the 1990s, and in recent years, researchers began to apply machine learning methods. This section sketches out the evolution of heuristics for the RCPSP and discusses their performance.

Priority rule heuristics. Papers like that of Kelley (1963) laid the foundation for methods that were based on two main components, a schedule generation scheme (SGS) and a priority rule. The SGS organizes the scheduling process. The serial SGS (s-SGS) selects an activity in each step and schedules it to start at the earliest time that is feasible with regard to the precedence and resource constraints. In contrast, the parallel SGS (p-SGS) is based on a current time which is incremented during scheduling, and in each step, an activity is selected to start at that time. Of course, the serial and the parallel SGS are the heuristic counterparts of the two branching schemes of the same names (see Section 6). The priority rule specifies the calculation of a priority for each activity and thus determines which activity is selected in each step of an SGS. Classical rules include, for example, the latest finish time rule (LFT) and the minimum slack rule (MSLK).

The two SGS differ in their search spaces and thus in their behavior (Kolisch, 1996). The parallel SGS outperforms the serial one when applied to larger projects with scarce resource capacities if the number of calculated schedules is rather small. This is because the parallel SGS tends to build compact schedules with high resource utilization. However, its search space is smaller than that of the serial SGS, and unlike the latter, it may in fact exclude all optimal solutions (Sprecher et al., 1995). Thus, the serial SGS is better suited to reach (near-)optimal schedules.

To improve the solution quality, priority rule heuristics were extended by randomizing the activity selection in the SGS, and priority rules were used to derive probabilities for the activities to be selected. These so-called sampling methods allowed to produce many schedules with a priority rule (and to pick the best one). Naturally, the stochastic multi-pass heuristics outperform their deterministic single-pass relatives and were state of the art in the mid-1990s (Kolisch, 1996). More recently, the automatic design of new priority rules to solve the RCPSP using genetic programming has been on the research agenda (Dumic & Jakobovic, 2021; Luo et al., 2022).

Metaheuristics. Multi-pass priority rule methods have a major limitation: They are unable to learn from previously calculated schedules. This motivated the scientific community to shift the focus towards the more intelligent metaheuristics. Nevertheless, priority rule heuristics remain relevant as they are employed to find initial solutions for metaheuristics, and also the SGS remain important building blocks of metaheuristics. Moreover, priority rule methods are still applied in commercial software because they are fast and intuitive.

Many metaheuristics require the modification of a solution or the recombination of two solutions to produce a new one. However, there are no straightforward operators that would produce feasible and efficient schedules for the RCPSP. Therefore, instead of operating directly on schedules, most metaheuristics operate on representations (or encodings) which are then transformed into schedules. Boctor (1996) was the first to present a simulated annealing heuristic for the RCPSP. He used an activity list (AL) representation which is turned into a schedule

by the serial SGS which, in each step, picks the next activity to be scheduled from the list. Hartmann (1998) compared three representations within a genetic algorithm (GA) in a computational study, and the activity list representation performed better than the random key and the priority rule representation.

The first and still fairly basic metaheuristics mentioned above were a starting point for many developments that took place since the late 1990s. Many new ideas have been developed, and metaheuristics have been fine-tuned in rigorous computational experiments. The main trends that have shaped the evolution of metaheuristics are the following:

- New and improved representations were developed. One goal was to fix the shortcomings of standard representations like the activity list and the random key representations. Both of the latter contain redundancy, that is, different representations may represent the same schedule. To overcome this issue, more advanced approaches were proposed, including, e.g., the so-called standardized random key representation of Debels et al. (2006), disjunctive graph inspired representations (Artigues et al., 2003; Artigues & Roubellat, 2000; Roux, 1998) and the event-list representation of Paraskevopoulos et al. (2012). Nevertheless, the classical activity list representation is still among the most used approaches according to the survey of Pellerin et al. (2020).
- While the basic operators in early metaheuristics relied on random decisions, more intelligent operators were designed to exploit problem-specific knowledge. An example is the peak crossover of Valls et al. (2008) that aims at inheriting blocks of activity lists that represent parts of schedules with high resource utilization. This idea was further extended by Debels and Vanhoucke (2007) who used the total resource utilization (TRU) to define crossovers in a decomposition-based genetic algorithm with two populations.
- The concept of self-adaptation was first applied to the RCPSP by Hartmann (2002). He included both SGS into a genetic algorithm and added a gene to the activity list representation that decides which of the two SGS is used for scheduling the related activity list. This way, the evolution will automatically prefer the SGS that is better suited for the given project instance and more successful during the current phase of the search. As noted by Pellerin et al. (2020), self-adaptation has been applied to various other parameters of metaheuristics in recent years.
- A particularly successful idea was to improve schedules by shifting activities to the right within the schedule and then to the left. This procedure, often referred to as forward–backward improvement (FBI) or justification, is very effective at reducing the makespan of a schedule. It was applied by Tormos and Lova (2001) within a sampling heuristic and tested by Valls et al. (2005) within several priority rule methods and metaheuristics and has become a very popular add-on to many recent heuristics.
- Many different metaheuristic paradigms have been applied to the RCPSP. Pellerin et al. (2020) mention more than 25 metaheuristic strategies which can be found in recent papers, and the list is still getting longer: Further metaheuristic principles are continuously added. A recent example is the chemical reaction optimization (CRO) framework studied by Shuvo et al. (2023).
- Many researchers suggested hybrid procedures that combine several metaheuristic concepts, or even exact and heuristic components, such as in the large neighborhood search method (Palpant et al., 2004). When MILP and/or branch and bound are the main components, such methods are called matheuristics (see Section 6). In fact, hybridization has been a major trend in recent years, and it is a successful one as it allows for a very effective search (e.g., with diversification and intensification phases). In a recent study, Vanhoucke and Coelho (2024) showed that implementing a matheuristic for the RCPSP involves numerous choices, making it a challenging endeavor. In their survey, Pellerin et al. (2020) provide a classification of hybridization strategies and report excellent results. This is in line with the findings of Van Peteghem and Vanhoucke (2014) who compare metaheuristics for the multi-mode extension of the RCPSP (cf. Section 9). They conclude as well that hybrid methods often yield better results than simple metaheuristics.

Machine learning. It can be argued that metaheuristics make use of machine learning principles because their decisions are based on learning strategies that are applied during the search process. In fact, the concepts behind metaheuristics are often categorized as reinforcement learning, due to the feedback induced by the objective function. Nevertheless, in recent years, researchers have developed new heuristic approaches for the RCPSP based on machine learning.

One approach is to apply machine learning to identify promising heuristic strategies for a given project instance, as already mentioned in Section 6. Guo et al. (2021) collected a training set consisting of projects with indicators reflecting the characteristics of each project (see Section 3) along with the results of various priority rules for each project. They used supervised learning to predict the best priority rules for projects with certain properties. The approach with the most promising results makes use of multi-label classification and decision trees. Guo et al. (2021) mention that their methodology is not restricted to the standard RCPSP or priority rules and can be extended to other problems and heuristics. Zhao et al. (2022) use deep reinforcement learning and more precisely the proximal policy approximation algorithm to train a graph neural network representing the parallel schedule generation scheme to design an efficient priority rule. Despite the good performance of these studies, it is important to note that such machine learning algorithms do not construct resource-feasible schedules from scratch, but rather select from existing scheduling methods to maximize the likelihood of building an optimal schedule.

There has already been a substantial amount written about the potential of machine learning in project management and control, but most studies remain limited to project time/cost/quality management (Uddin et al., 2023), time forecasting (Wauters & Vanhoucke, 2014, 2016) or cost forecasting (Narbaev et al., 2024; Unsal-Altuncan & Vanhoucke, 2024), but rarely if ever extending to the construction of an optimal resource-feasible project schedule for the RCPSP. Hence, to the best of our knowledge, there are almost no studies using a machine learning methodology to directly solve the RCPSP by constructing a schedule themselves, although it is expected that this may change with recent developments. As a notable exception, Teichteil-Königsbuch et al. (2023) used a training set of project instances along with the optimal schedules obtained with a solver. They applied supervised learning to find a schedule for a given project instance. This was achieved by a specific graph representation of the instances (in which also the resources are included). A graph neural network which can handle input with such a graph structure was used. The schedules constructed by the graph neural network contained violations of the precedence and resource constraints. Therefore, the order of the activities was extracted from the infeasible schedules, and the serial SGS was used to determine feasible schedules from the activity list. The previously mentioned study by Luo et al. (2022) also relied on existing data and solutions for the RCPSP to train a genetic programming algorithm in an attempt to schedule new instances for the RCPSP. Dumic and Jakobovic (2021) followed a similar approach and integrated further machine learning techniques like bagging and boosting into the genetic programming framework. However, the quality of the resulting schedules is likely to remain relatively low because these studies rely on priority rules.

Table 2
Results of selected heuristics for PSPLIB set with 120 activities (Average deviations in % from critical path lower bound).

Heuristic	Paper	max. #schedules		
		1000	5000	50,000
Hybrid GA+local search, non-standard repr.	Proon and Jin (2011)	33.45	31.51	30.45
Hybrid scatter+local search, event list, s-SGS	Paraskevopoulos et al. (2012)	33.32	32.12	30.78
Hybrid GA+decomposition, FBI, TRU	Debels and Vanhoucke (2007)	34.19	32.34	30.82
Hybrid CRO+GA, AL, both SGS	Shuvo et al. (2023)	33.85	32.42	30.95
Hybrid GA+FBI, AL, s-SGS, peak crossover	Valls et al. (2008)	34.07	32.54	31.24
Hybrid scatter search+electromagn.+FBI	Debels et al. (2006)	35.22	33.10	31.57
Hybrid GA+FBI, AL, s-SGS	Valls et al. (2005)	35.39	33.24	31.58
GA, AL, both SGS, self-adapting	Hartmann (2002)	37.19	35.39	33.21
GA, AL, s-SGS	Hartmann (1998)	39.37	36.74	34.03
sampling, LFT, both SGS, FBI	Tormos and Lova (2001)	36.49	35.81	35.01
sampling, LFT, p-SGS	Kolisch (1996)	39.60	38.75	37.74
sampling, LFT, s-SGS	Kolisch (1996)	42.84	41.84	40.63

Performance. The evaluation of the performance of heuristics was arguably facilitated by two developments in the mid to late 1990s. First, the PSPLIB dataset of Kolisch and Sprecher (1997) described in Section 3 provided challenging project instances for computational studies. Second, Hartmann and Kolisch (2000) proposed an easy-to-use criterion as a basis for comparisons. Of course, a comparison of heuristic results is only fair if the computational effort to achieve the results is more or less the same in all heuristics to be compared. However, measuring the computational effort by CPU time would be misleading if the results are obtained with different programming languages and programming skills and, most importantly, on different computers. Thus, Hartmann and Kolisch (2000) suggested to use the number of schedules that were calculated for an instance as a measure of the computational effort. This idea is based on the assumption that building one schedule within any heuristic involves the calculation of a start time for each activity as done in one pass through an SGS. Therefore, the number of schedules calculated for a project reflects the computational effort—not perfectly, but in a pragmatic way. The first approach was to report the results of a heuristic after 1000 and 5000 schedules were constructed for a project instance, respectively. Increasing computer power allowed for the computation of more schedules in reasonable time, which motivated Kolisch and Hartmann (2006) to add a limit of 50,000 schedules. Overall, this allowed all researchers to compare the results of their heuristics very easily to results from the literature. Of course, researchers are responsible for applying the stopping criteria correctly when testing their heuristics.

The most recent study comparing the performance of a large number of heuristics proposed in the literature is that of Pellerin et al. (2020). They show that the performance of today's advanced metaheuristics is far above that of the first metaheuristics for the RCPSP that appeared in the late 1990s. Overall, an impressive number of heuristics have been developed to this day, and the computational results have steadily improved. However, the study of Pellerin et al. (2020) also shows that the solution quality did not improve that much anymore during the last decade.

To give an impression of the development regarding the performance of heuristics, Table 2 provides the computational results of several heuristics for the PSPLIB dataset with 120 activities. To keep the table short, it is restricted to those heuristics cited in this section for which results were available, and it adds the hybrid metaheuristic of Proon and Jin (2011) which is the best performing heuristic for this dataset in the study of Pellerin et al. (2020). Since many optimal solutions are unknown, results are given as percentage deviations from the critical path based lower bound (which is not a tight bound but easy to use and thus widely accepted). The purpose of the table is to sketch out the progress achieved in the last decades. Of course, there are many other successful heuristics. A comprehensive comparison can be found in Pellerin et al. (2020).

We believe that further research on the topics listed above is worthwhile and will probably lead to even better results in the future. Our impression is that problem-specific ideas for the representation and the operators as well as smart hybridization and machine learning approaches may be more promising directions for research than the metaheuristic paradigms themselves. It would also be interesting to obtain more optimal solutions or tighter lower bounds to judge the current solution gaps.

Overall, recent developments have led to extremely powerful but also considerably complex heuristics. However, as Pellerin et al. (2020) mention, simplicity should also be valued because simpler heuristics may be more attractive for practitioners. Moreover, heuristics for the standard RCPSP are often used as starting points when developing heuristics for extended project scheduling problems. In this regard, simpler heuristics might be more straightforward starting points.

As sketched out above, the availability of challenging datasets and easy-to-use criteria for comparison have led to a very competitive environment for research on heuristics for the RCPSP (e.g. Coelho & Vanhoucke, 2023 proposed the so-called sD dataset especially for testing new metaheuristic procedures). Of course, this has contributed to the excellent results. We would like to add that the race for even better heuristic results should not necessarily be the only goal. Fundamentally new ideas are also valuable even if they do not outperform most other heuristics right away. Also, gaining deeper insight into what makes a component of a heuristic successful and why (and possibly for what kinds of project instances) would be an important goal for future research. We think that experiments that analyze the impact of the components of heuristics on the performance deserve more attention.

8. Lower bounds

Lower bounds are of multiple interests for an NP-hard optimization problem like the RCPSP. First, as stated in Section 6, lower bounds are used to prune nodes in branch-and-bound trees. For this usage, fast lower bounds are needed since they are computed thousands of times. Second, Section 7 revealed the vast amount of research that has been carried out on heuristic and metaheuristic approaches due to both the practical interest of the problem and its intractability. To evaluate the quality of various heuristics, a comparison with the exact solution value can only be made on small instances. Having strong lower bounds is a means to have a tight estimation of the optimality gap of the compared heuristics. In this case, more time can be spent for obtaining them. A last usage is to use the solution of the relaxed problem to guide the search towards a feasible solution. As well explained in Klein and Scholl (1999), there are two ways of obtaining a lower bound on the optimal makespan of the RCPSP. The constructive way solves a relaxation of the RCPSP makespan minimization problem whose optimal value gives a lower bound. The destructive way solves a relaxation of the decisional RCPSP for a given makespan T. In case of proven infeasibility a lower bound of value $T + 1$ is obtained.

The LP relaxation of all the (mixed-)integer linear programs presented in Section 4 can be used as constructive lower bounds. A

destructive lower bound can also be obtained by fixing the makespan to the threshold value and solving the LP relaxation or, for even better bounds, the MILP formulation to tentatively prove infeasibility in a destructive approach. Similarly, all the consistency tests presented in Section 5 work on a relaxation of the decisional RCPSP and can (and have been) used to compute destructive lower bounds.

In various papers, the makespan obtained by the compared heuristics is given as a percentage of the critical path lower bound (see also Section 7). However, much better lower bounds can be obtained by using constructive or destructive LP and CP-based lower bounds or other dedicated lower bounds. Many of them were used to prove the optimality of solutions found by heuristics in benchmark instances from the literature. Surveys on lower bounds for the RCPSP can be found in Demassey (2013), Demeulemeester and Herroelen (2006), Knust (2015), Néron (2008), Neron et al. (2006). We list below different lower bounding approaches either based on above-described LP and CP models or on other relaxations.

In the early years, some proposed lower bounds were based on the constraint propagation principle before this concept was even defined. For example, this is the case for the basic resource bound, giving the maximum ratio (rounded to the upper integer) of the total resource requirement on a resource divided by its availability, which can also be obtained destructively by the cumulative overload check on interval $[0, T]$ where T is the time horizon. This is also the case for the lower bounds by Stinson et al. (1978), Zaloom (1971) that both improve the critical path lower bound by energetic reasoning considerations. The first bound considers the cumulated requirement on a resource k over time of either the earliest or the latest start schedule given an upper bound. A valid lower bound is then the smallest value LB such that at each time t, the cumulated requirement stays below $B_k t$. Hence, we have a sort of premises of energetic reasoning by proving infeasibility of right shifts or left shifts in a sliding interval. Second, the critical sequence bound considers the resource profile associated with the activities on a critical path of length L. If an activity i can be scheduled in parallel with the critical path due to resource requirement during interval of maximal length $[a, b]$ where $[a, b]$ is included in the activity time window $[ES_i, LC_i]$ then $L+b-a$ is a valid lower bound. Demeulemeester and Herroelen (1992) extend this bound to a path of non-critical activities via dynamic programming. A fast bound, the weighted node packing bound, was proposed by Mingozzi et al. (1998) issued from the dual of a preemptive relaxation of the feasible set MILP formulation (see Section 4). The dual correspond to a weighted node packing problem, an NP-hard problem but any upper bound for the dual gives a feasible lower bound for the primal, which allows to obtain a fast lower bound of reasonable quality.

Constructive variants of the consistency tests based on relaxations of the cumulative constraint (see Section 5) can be used to obtain fast lower bounds. The due dates d_i of activities in the cumulative constraints are replaced by tails q_i where the tail can be initialized to the length of the longest path from the activity to the dummy end node. We obtain the optimization variant of the cumulative constraint, called the cumulative scheduling problem (CuSP) $\min \max_{i \in \mathcal{A}} S_i + p_i + q_i$ subject to release date and resource constraints. Parallel machine relaxations of the CuSP can the be obtained by techniques detailed in Carlier and Latapie (1991). Then, given a parallel machine problem with release dates and tails, the most common lower bound is the subset bound (Perregaard, 1995; Vandevelde et al., 2005) which search for the subset $I \in \mathcal{A}$ such that maximizes $1/m(r_{i_1} + \cdots + r_{i_m} + \sum_{i \in I} p_i + q_{j_1} + \cdots + q_{j_m})$, where $i_1, \ldots, i_m$ are the m smallest release dates of activities in I and $j_1, \ldots, j_m$ are the m smallest tails in I. The bound can be computed in $O(|\mathcal{A}|^2)$. The pseudo-preemptive relaxation of the CuSP can be solved in $O(|\mathcal{A}|^2)$ by the Jackson's pseudo-preemptive scheduling algorithm (Carlier & Pinson, 2004). In Carlier et al. (2023) the authors propose fast constructive variant of energetic reasoning with $O(|\mathcal{A}| \log |\mathcal{A}|)$ and $O(|\mathcal{A}|^2)$ time complexity. The above-described bounds have the advantage of being fastly computable enough for being integrated in a branch-and-bound method such as in Demeulemeester and Herroelen (1992, 1997a) for the critical sequence bound and the weighted node packing bound and in Carlier and Latapie (1991) for the subset bound. However, when many bounds are used, the earlier discussed credit system (see Section 6) used in the branch-and-bound method of Coelho and Vanhoucke (2018) assures that not too much time is spent in calculating their values.

When the objective is rather to obtain a near-optimal lower bound disregarding the computation times, exact CP (Section 5) and MILP (Section 4) also produce tight lower bounds. Two representatives of this category are the lazy clause generation CP/SAT method by Schutt et al. (2011) and the branch and cut method by Araujo et al. (2020) that both were able to find new lower bounds on hard instances from the PSPLIB. Exact MILP solving of NP-hard relaxations is also a way to obtain good lower bounds, such as in Moukrim et al. (2015) where the preemptive RCPSP is solved by branch and cut with a formulation close to the Mingozzi et al. (1998) formulation, and the RCPSP with aggregated resource constraint (Morin et al., 2022) solved by MILP formulations. Carlier and Néron (2000) present the multiple elastic relaxation of the CuSP where at each time period, the amount of resource allocated to an activity is a multiple of its resource requirement. The relaxation can be solved by a linear program based on the enumeration of feasible configurations given the resource capacity B_k. The LP can be solved fastly up to $B_k = 11$ by a parametric approach. Haouari et al. (2014) extend the method proposed by Carlier and Néron (2003) to obtain destructive lower bounds by solving a relaxation of the preemptive RCPSP including pseudo precedence constraints by linear programming. Haouari et al. (2014) include new valid inequalities. The approach proved to be competitive with the lower bounds obtained by Schutt et al. (2011), Vilím (2011). With the objective to obtain tight lower bounds faster than the above strong but time-consuming approaches, Arkhipov et al. (2019) proposed a pseudo-polynomial algorithm to compute lower bounds on the consecutive evaluation of pairs of resources and their cumulated workload.

Using relaxations to guide the search towards promising solutions has been hardly explored in the RCPSP literature. Möhring et al. (2003) use a Lagrangian relaxation of the resource constraints on the $P-DDT$ formulation (see Section 4) to solve the LP relaxation via minimum cut computations and obtain good heuristic solutions by using the relaxed solution inside a priority-rule based heuristic.

Joining the forces of MILP and CP to obtain tight destructive lower bounds have been successful, either by exploiting the time-indexed and continuous time sequencing formulation using cutting planes issued from CP (Demassey et al., 2005) or by alternating column generation on the Mingozzi et al. (1998) formulation and constraint propagation as in Baptiste and Demassey (2004), Brucker and Knust (2000a, 2003), Demassey et al. (2004). Under the light of the recent breakthrough in MILP and CP lower bounds, this research area deserves to be further explored towards CP/MILP/SAT approaches.

9. RCPSP variants

The standard RCPSP is a very lean and minimalist problem, but despite its simple structure, it is very hard to solve. One may argue that this has contributed to its success: A clearly defined, simple problem that is challenging from a computational point of view provides an attractive playground for developing exact and heuristic algorithms. Nevertheless, from a more application-oriented point of view, the RCPSP assumptions are too strict to capture properties and requirements of many real-world projects. Therefore, more detailed problem settings are needed for scheduling projects in practice.

Many more complex project scheduling problems have been discussed in the scientific literature, and the RCPSP has often been used as a starting point by varying and extending its assumptions, restrictions, and objective function. An overview of extended models has been given in Hartmann and Briskorn (2010), and an update to cover the next

decade followed in Hartmann and Briskorn (2022). In this section, we summarize the most important variants and extensions. We proceed by looking at generalizations of the activity, resource, and precedence constraints first and then continue with a look at alternative objectives and models for multiple projects. We also briefly mention studies that considers uncertainty in a stochastic and/or robust setting.

Activities. In the basic RCPSP, an activity is related to a fixed non-preemptable duration and constant amounts of one or more resource types that are required during the duration. The concept of multiple modes dates back to Elmaghraby (1977) and allows for more flexibility by introducing several ways to accomplish an activity: Each mode is defined by a duration and requests for resources, which permits, for example, to choose between alternative resources for an activity and to shorten its duration by using more resource units per period. For each activity, not only a start time but also a mode has to be selected. If nonrenewable resources are added (see below), we obtain the multi-mode RCPSP (MRCPSP) which has become a popular standard problem in its own right. If more than one nonrenewable resource is given, the feasibility problem is NP-complete (Kolisch & Drexl, 1997). A computational comparison of branch-and-bound methods is given by Hartmann and Drexl (1998) while Van Peteghem and Vanhoucke (2014) compare heuristics. An in-depth literature review of the MRCPSP and related problems is given in Weglarz et al. (2011).

A second way to extend the structure of activities is to drop the assumption that resource usage is constant during the duration. This can be achieved by allowing for time dependent resource requests b_{ikt}. This reflects the demand of activity i for resource k in the tth period of its duration. Hartmann (2013) shows that this can be useful in medical research projects. A different approach is to consider a fixed workload of an activity but to leave it to the scheduling process to decide how many units of a resource are allocated to the activity in each period. Often, further requirements such as a minimum and a maximum per-period request are added. This approach is often referred to as flexible resource profile (Naber & Kolisch, 2014), and the idea has been included into several otherwise different problem settings (Bianco & Caramia, 2011; Fündeling & Trautmann, 2010).

Another generalization of the activity concept of the RCPSP is to relax the constraint that an activity may not be interrupted once it has started. Interruptions are often only allowed at integer points in time. This leads to the so-called preemptive RCPSP. Several researchers have added further constraints to limit the interruptions. For example, in Quintanilla et al. (2015), each activity is associated with a limit on the number of interruptions, and each resulting part of an activity must have a minimum duration. Finally, setup times between two activities have been considered by several researchers. Vanhoucke and Coelho (2019) consider setup times that are taken into account when an interrupted activity is continued. Closely related to setup times are transfer times which occur if a resource has to be transported from the location of one activity to that of another activity (Krüger & Scholl, 2010).

Temporal constraints and network. Next, we focus on the temporal conditions induced by the precedence relations of the RCPSP and how they can be generalized. In the basic RCPSP, an activity is only allowed to start when its predecessors have finished. This corresponds to finish–start constraints. Analogously, start–start, start–finish, and finish–finish constraints could be defined, but as shown by Bartusch et al. (1988), these types can be transformed into finish–start relations (this holds for the classical RCPSP but not for some extensions like the MRCPSP). Also, time-lags can be introduced, indicating the minimal and maximal time that is allowed to pass between two activities. Adding these time lags to the RCPSP leads to an extended model that is often referred to as RCPSP/max. Obviously, the standard RCPSP contains minimal time lags of 0. Bartusch et al. (1988) have shown that the general time lags include release dates and deadlines for activities and even time-dependent resource requests of the activities as special cases. They have also proven that the feasibility problem of the RCPSP/max is NP-complete. Dorndorf et al. (2000c) have proposed a branch-and-bound method for the RCPSP/max.

The temporal constraints can also be extended by allowing activities to overlap, that is, to be processed in parallel for some time. One idea is to allow an activity to start when its predecessor has completed a certain percentage of its workload. There are further variants such as, for example, allowing an activity to finish when its predecessor has completed a certain percentage of its workload. These so-called feeding precedence constraints have been employed by several researchers (e.g., Bianco & Caramia, 2011), often together with a concept similar to the flexible resource profiles mentioned above. A different approach to overlapping was proposed by Hartmann (2013) where it can be specified that two activities do not start at the same time, do not finish at the same time, or do not overlap at all. These temporal constraints differ from the usual precedence relations in that they do not impose an order of the activities. Vanhoucke and Coelho (2016) further add a so-called changeover time that must pass between two activities that may not overlap (but for which no order is prescribed). They also add a new type of precedence constraint according to which an activity may already start when at least one predecessor has finished.

In usual precedence networks, all activities must be carried out. Several researchers extended the concept of networks by incorporating the decision which activities will actually be executed and which will not, which leads to a flexible project structure. In the RCPSP, each activity can be viewed as a logical "AND" node, meaning that all its successors must be carried out. In an extended network, an activity can also be an "XOR" or an "OR" node, which implies that exactly one or at least one immediate successor must be executed. The models suggested in the literature differ in their assumptions, and there is no standard yet. An example is the approach of Servranckx and Vanhoucke (2019) who include so-called principal activities that represent "XOR" nodes and trigger the selection of exactly one subgraph of activities. A similar concept is suggested in der Beek et al. (2024).

Resources. Let us now take a brief look at variants of the resource concept. The first one is to replace the constant resource capacities of the basic RCPSP with time-dependent ones. Hartmann (2013) combines these time-varying capacities with time-varying resource requests (see above) and shows that the serial SGS loses its property to always include an optimal schedule in the search space when applied to this extended setting.

In addition to renewable resources, various other resource categories have been proposed. A classic category is that of nonrenewable resources (Slowinski, 1980, 1981) which are limited for the entire project, unlike the renewable resources which have a limited per-period supply. Nonrenewable resources are usually considered when multiple modes are given (see above). Partially renewable resources are limited for individual sets of time periods, which makes them a very general category as they include both renewable and nonrenewable resources as special cases. A recent study that employs partially renewable resources (and maximal time lags) was presented by Watermeyer and Zimmermann (2023).

Storage resources[8] (Neumann & Schwindt, 2002) are used by some activities and produced by others. Throughout the project, the inventory level of a storage resource must usually be between a lower and an upper limit. Recently, der Beek et al. (2024) used storage resources in an RCPSP with a flexible project structure. A related category often called material reflects resources that are consumed by activities and have to be ordered from external suppliers, with the timing and quantities of the orders being part of the schedule to be determined (e.g., Fu, 2014).

[8] sometimes also called cumulative resources, see our disambiguation note page 1.

Finally, resources with multiple skills have become a popular ingredient in project scheduling problems. In the multi-skill RCPSP (MSRCPSP) that has become a widely accepted standard, a number of skills is given, and for each (usually human) resource, it is known whether it masters a skill or not. Each activity requires resources with certain skills. This setting has been extended by various researchers. For example, some models include levels to express how well a resource masters a skill, and activities now require resources with skills at or above a certain level. Snauwaert and Vanhoucke (2023) give an overview of the MSRCPSP and its variants and extensions.

Objectives. Many different objectives have been considered for resource-constrained project scheduling. In what follows, we discuss the most popular types of objectives. The objective of the standard RCPSP, the minimization of the makespan, belongs to the category of time-based objectives. Another popular time-based objective is to minimize the weighted sum of earliness and tardiness with regard to given due dates for the activities (Vanhoucke et al., 2001a). Of course, it includes the also frequently considered weighted tardiness objective as a special case.

Projects do not always proceed as planned. For example, activities may take longer than expected or resources may become temporarily unavailable. In this context, time-based objectives play an important role as well. If deviations from the original schedule have occurred while the project is in progress, the project may have to be rescheduled based on the updated data. If the new schedule should stick as closely as possible to the original one, the original activity finish times can be used as due dates, and the objective to minimize weighted earliness and tardiness can be used for rescheduling (of course, the start times of activities already finished or in progress cannot be changed). In addition to this reactive approach, a proactive strategy is to include slack times into the schedule to make the latter more robust without considering explicitly stochastic models nor uncertainty scenarios. Possible delays of activities can then be at least partially absorbed by the slack times, which reduces the risk of a delayed completion of the project. Several slack-related objectives have been proposed, usually within a model with a deadline to fix the project duration. Kobylanski and Kuchta (2007) suggest to maximize the smallest free slack, either as an absolute value or in relation to the activity duration. Chtourou and Haouari (2008) compare several surrogate robustness measures, including total slack of the activities and maximum number of activities with slack. Also, both measures are extended by weights for the activities.

Next, we look at resource-based objectives. In models with resource-related objectives, the makespan is often bounded by a deadline constraint. Among the most popular objectives in this category is the minimization of the weighted resource capacities (the weights can reflect unit availability costs of the resources). The RCPSP with this objective and with a deadline constraint is often referred to as resource availability cost problem or resource investment problem. It has been tackled my many researchers, see, e.g., Van Peteghem and Vanhoucke (2015). Closely related are resource leveling objectives which seek to minimize changes in the resource profile. Several different resource leveling objectives have been proposed, including the weighted absolute or squared period-to-period changes in the resource profile as well as the weighted squared per-period requests (see, e.g., Ponz-Tienda et al., 2017). Another related objective is the minimization of the total resource overload, that is, the minimization of the weighted utilization of the resources that is above a given threshold (e.g., Rieck et al., 2012).

Taking into account cash flows during a project's execution leads to another type of objective. The execution of activities and utilization of resources result in cash outflows, whereas cash inflows occur upon the completion of project milestones. In projects with a longer horizon, payments are discounted using an interest rate. The resulting objective is to maximize the net present value (NPV) of the project (e.g., Vanhoucke et al., 2001b and Leyman & Vanhoucke, 2017). Again, a deadline is added to limit the duration of the project. The standard RCPSP with discounted cash flows can be further extended. For example, Leyman and Vanhoucke (2017) add a constraint that the capital cannot be negative in any period, and they allow for different timings of payments during the execution of the activities. Yet another approach is followed by Khoshjahan et al. (2013) who maximize the NPV in a model with due dates where cash flows are caused by earliness-tardiness penalties, while Nikoofal Sahl Abadi et al. (2018) discount costs caused by changes in the resource profile over time.

In some situations, it may be too limiting to consider only a single objective, which leads to a multi-objective approach (Slowinski et al., 1994). There are two main ways to handle multiple optimization criteria. A straightforward approach is to combine the criteria within one objective function using weights for the criteria, which essentially reduces to a single-objective approach. An example is the model of Van Peteghem and Vanhoucke (2015) who combine a resource-based and a time-based objective by minimizing the sum of resource availability costs and tardiness costs. Another way to deal with multiple objectives is to determine a set of Pareto-efficient schedules with regard to the objectives. An example for this case is Nikoofal Sahl Abadi et al. (2018) who consider both the makespan objective and a resource leveling objective. Ballestín and Blanco (2015) discuss fundamental aspects of multi-objective project scheduling.

Multiple projects. While all models discussed so far are designed for scheduling single projects, it is easy to adapt them such that multiple projects can be scheduled within one integrated model. In fact, this makes sense if different projects utilize the same set of resources. The activity networks of the individual project can simply be combined into one "super network". Together with the usual constraints of the standard RCPSP and possibly release dates for the projects, we obtain the basic resource-constrained multi-project scheduling problem (RCMPSP). Common objectives of the RCMPSP are, for example, the minimization of the total portfolio makespan (i.e., the duration until all activities of all projects are finished), the minimization of the average project makespan (which takes the makespans of the individual projects into account), and the minimization of the average project delay (i.e., the average tardiness of the projects with regard to their due dates), see Bredael and Vanhoucke (2023).

The basic RCMPSP as described above is the core of most multi-project scheduling approaches. Many variations and extensions have been employed, including some that are used in single-project models as well. Local and global resources are of particular relevance in the context of multi-project scheduling. Whereas local resources are available only for a specific project, global resources are shared among projects. Local and global resources are considered by, e.g., Adhau et al. (2013) who further extend this concept by adding transfer times that apply when a global resource is moved from the location of one project to that of another. Another important feature of some multi-project problems is the decision which of the projects should be selected to be executed. This setting has been tackled by, e.g., Shariatmadari et al. (2017). For an in-depth review of variants of the RCMPSP, we refer to Gómez Sánchez et al. (2023). Studies of priority rule based heuristics and metaheuristics for the basic RCMPSP can be found in Browning and Yassine (2010) and Bredael and Vanhoucke (2023), respectively. Just as was the case for the single-project RCPSP (cf. Section 3), datasets for the multi-project problem have been proposed by Homberger (2007), Vázquez et al. (2015), Browning and Yassine (2010), Van Eynde and Vanhoucke (2020, 2022a).

Uncertainty. So far we limited ourselves to deterministic settings. But real life projects are subject to various sources of uncertainty that affect activities (presence, duration, resource requirements, ..) and resources (availability, speed...). A large amount of research has been devoted since decades to the management of uncertainty in the RCPSP and its variants and reviewing this research is beyond the scope of this survey. They can be roughly divided in proactive approaches where a baseline

schedule is built to hedge against some apriori knowledge on uncertainty and reactive approaches where a reactive policy is designed to build or repair the schedule when unexpected events occur. Stochastic programming with recourse, chance-constrained programming, Markov decision processes, fuzzy optimization and robust discrete optimization are the most widely encountered techniques to design such proactive-reactive methods. For the interested reader we refer to state-of-the-art surveys (Demeulemeester & Heerroelen, 2002, 2013; Demeulemeester et al., 2010; Herroelen & Leus, 2005; Schwindt et al., 2015; Ulusoy & Hazır, 2021).

Naturally, the above summary of extensions and variants of the RCPSP is far from complete. Researchers are proposing an ever-inc reasing number of new features which cannot be presented in more detail on a few pages. Nevertheless, we tried to point to some important developments. The survey papers cited throughout this section provide links to hundreds of papers that are worth exploring. Several RCPSP variants have become established standards in their own right. This includes the multi-mode RCPSP, the RCPSP with minimal and maximal time-lags, the multi-skill RCPSP, the resource availability cost problem, the RCPSP with discounted cash flows, and the basic multi-project RCPSP. Often, a competitive environment developed, with researchers comparing their results for a standardized problem. On the other hand, the design of new assumptions, constraints, and objectives beyond standard models is an important part of the project scheduling evolution as well. It helps to create features and models which are of high practical relevance. In fact, motivating new concepts by actual real-world applications is valuable for the scientific community, and, therefore, deserves more research attention.

10. Conclusion

This article has provided a unique, but incomplete, overview of the research on the challenging problem of resource-constrained project scheduling. It was shown that the problem has many theoretical challenges, but is also interesting for further research from a computational point of view. Many mixed integer programming models, branch-and-bound procedures and recently a growing number of meta-heuristic methods have been developed, and therefore much progress has been made. Presenting project data in the academic literature was a good help to test (and compare) the different algorithms, which ultimately led to powerful procedures that can solve projects to near-optimality. We also presented software packages dedicated to the RCPSP issued from the mentioned research on local search or constraint programming but we omitted the integration of the RCPSP algorithms in project management software. The production of such software is rapidly increasing and reviewing them is out of the scope of this paper. More or less recent studies reveal that mainly simple heuristics are included in such software (Albayati & Aminbakhsh, 2023; Baumann & Trautmann, 2015, 2016; Trautmann & Baumann, 2009) illustrating the gap between research and practice in project management and scheduling that has yet to be closed (Scales, 2020).

Promising avenues of research can be highlighted from this state of the art review. About complexity and approximation analyses of the RCPSP, further research on parameterized complexity and in particular on fixed-parameter tractability appears as particularly appealing as underlined by Ganian et al. (2020). About exact solution approaches, tremendous progress has been achieved independently by mixed-integer linear programming through extended formulations and strong valid inequalities on one side and constraint programming/boolean satisfiability via conflict-directed search, global constraint explanations and clause learning on the other side. Reaching a new breakthrough in exact approaches potentially needs to join and unify the research on MILP, CP, SAT and SMT approaches to design hybrid exact methods, e.g. by deriving SAT clauses from strong valid inequalities or vice versa. We also underline the importance of empirical project data calibration with the rise of data-driven and machine learning approaches. The effective use of machine learning in the RCPCP is still in its infancy despite the reported progress, especially in the design of heuristics. Beyond the standard deterministic RCPSP, data-driven and machine learning approaches will more certainly be of primordial help to deal with complex variants and uncertainty.

Yet, despite this impressive progress, the problem is still challenging for further research. First and foremost, there is the observation that large projects still cannot be scheduled very well. Despite strong advances in computing power, there are still artificial project instances with only 60 activities for which no optimal solution has been found. Moreover, as researchers we still do not fully understand why one project instance can be solved (to optimality) while another cannot, and more research in the search for drivers of the problem complexity is certainly welcome. In addition, there is also the observation that the RCPSP has many possible extensions. Not only is minimizing the makespan a strong simplification (and can be extended to other objective functions), but the problem can also be made much more realistic by adding a variety of other features to the basic formulation of the problem (for which several have been mentioned in the article).

Finally, the problem also has many applications outside the domain of project management, where the RCPSP can form a basis for solving very different problems. Although a project is typically viewed as a temporary endeavor, more and more project scheduling concepts are being applied to repetitive environments, ranging from scheduling patient appointments in hospitals (Riise et al., 2016) and assigning aircrafts to gates at airports (Dorndorf et al., 2007) to handling arrivals of ships at ports (Hill et al., 2019) and the planning of table tennis competitions (Knust, 2010). This once again underlines the relevance of the RCPSP (and variants) and shows that the problem is so multi-faceted that the end of the research is not even close.

CRediT authorship contribution statement

Christian Artigues: Writing – original draft, Writing – review & editing. **Sönke Hartmann:** Writing – original draft, Writing – review & editing. **Mario Vanhoucke:** Writing – original draft, Writing – review & editing.

Acknowledgments

This work was partially funded by ANITI IA Cluster, Université de Toulouse. The authors warmly thank Roger Kameugne for having corrected the constraint programming section, as well as Alessandro Agnetis, Jean-Charles Billaut and Dvir Shabtay for their advices and corrections on complexity analysis and approximation.

References

Adhau, S., Mittal, M., & Mittal, A. (2013). A multi-agent system for decentralized multi-project scheduling with resource transfers. *International Journal of Production Economics*, *146*(2), 646–661.

Aggoun, A., & Beldiceanu, N. (1993). Extending CHIP in order to solve complex scheduling and placement problems. *Mathematical and Computer Modelling*, *17*(7), 57–73.

Albayati, N. H. F., & Aminbakhsh, S. (2023). Resource allocation capabilities of commercial project management software packages for resource leveling and resource constrained project scheduling problems: a comparative study. *Journal of Construction Engineering, Management & Innovation*, *6*(2), 104–123.

Alvarez-Valdes, R., & Tamarit, J. (1993). The project scheduling polyhedron: Dimension, facets and lifting theorems. *European Journal of Operational Research*, *67*(2), 204–220.

Ansótegui, C., Bofill, M., Palahı, M., Suy, J., & Villaret, M. (2011). Satisfiability modulo theories: An efficient approach for the resource-constrained project scheduling problem. In *Proceedings of the 9th symposium on abstraction, reformulation and approximation (SARA 2011)* (pp. 2–9).

Applegate, D., & Cook, W. (1991). A computational study of the job-shop scheduling problem. *ORSA Journal on Computing*, *3*(2), 149–156.

Araujo, J. A., Santos, H. G., Gendron, B., Jena, S. D., Brito, S. S., & Souza, D. S. (2020). Strong bounds for resource constrained project scheduling: Preprocessing and cutting planes. *Computers & Operations Research*, *113*, Article 104782.

Arkhipov, D., Battaïa, O., & Lazarev, A. (2019). An efficient pseudo-polynomial algorithm for finding a lower bound on the makespan for the resource constrained project scheduling problem. *European Journal of Operational Research*, *275*(1), 35–44.

Artigues, C. (2008). The resource-constrained project scheduling problem00. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 21–35.

Artigues, C. (2017). On the strength of time-indexed formulations for the resource-constrained project scheduling problem. *Operations Research Letters*, *45*(2), 154–159.

Artigues, C., Demassey, S., & Neron, E. (2013). *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons.

Artigues, C., Koné, O., Lopez, P., & Mongeau, M. (2015). Mixed-integer linear programming formulations. In C. Schwindt, J. Zimmermann, & et al. (Eds.), *Handbook on project management and scheduling vol. 1* (pp. 17–41). Springer,

Artigues, C., Michelon, P., & Reusser, S. (2003). Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, *149*(2), 249–267.

Artigues, C., & Roubellat, F. (2000). A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research*, *127*(2), 297–316.

Balas, E. (1969). Machine sequencing via disjunctive graphs: an implicit enumeration algorithm. *Operations Research*, *17*(6), 941–957.

Ballestín, F., & Blanco, R. (2015). Theoretical and practical fundamentals. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on project management and scheduling vol. 1* (pp. 411–427). Springer.

Baptiste, P., & Bonifas, N. (2018). Redundant cumulative constraints to compute preemptive bounds. *Discrete Applied Mathematics*, *234*, 168–177.

Baptiste, P., & Demassey, S. (2004). Tight LP bounds for resource constrained project scheduling. *Or Spectrum*, *26*, 251–262.

Baptiste, P., Le Pape, C., & Nuijten, W. (1999). Satisfiability tests and time-bound adjustmentsfor cumulative scheduling problems. *Annals of Operations Research*, *92*, 305–333.

Baptiste, P., Le Pape, C., & Nuijten, W. (2001). *vol. 39*, *Constraint-based scheduling: applying constraint programming to scheduling problems*. Springer Science & Business Media.

Baptiste, P., & Pape, C. L. (2000). Constraint propagation and decomposition techniques for highly disjunctive and highly cumulative project scheduling problems. *Constraints*, *5*(1–2), 119–139.

Bartusch, M., Möhring, R. H., & Radermacher, F. J. (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, *16*, 201–240.

Batselier, J., & Vanhoucke, M. (2015). Construction and evaluation framework for a real-life project database. *International Journal of Project Management*, *33*, 697–710.

Baumann, P., & Trautmann, N. (2015). Resource-constrained project scheduling with project management information systems. *Handbook on Project Management and Scheduling Vol. 2*, 1385–1400.

Baumann, P., & Trautmann, N. (2016). A note on the selection of priority rules in software packages for project management. *Flexible Services and Manufacturing Journal*, *28*, 694–702.

der Beek, T. V., Souravlias, D., van Essen, J., Pruyn, J., & Aardal, K. (2024). Hybrid differential evolution algorithm for the resource constrained project scheduling problem with a flexible project structure and consumption and production of resources. *European Journal of Operational Research*, *313*(1), 92–111.

Beldiceanu, N., Bourreau, E., Rivreau, D., & Simonis, H. (1996). Solving resource-constrained project scheduling problems with CHIP. In *Fifth international workshop on project management and scheduling, poznan, Poland*.

Bell, C. E., & Park, K. (1990). Solving resource-constrained project scheduling problems by a* search. *Naval Research Logistics*, *37*(1), 61–84.

Berthold, T., Heinz, S., Lübbecke, M. E., Möhring, R. H., & Schulz, J. (2010). A constraint integer programming approach for resource-constrained project scheduling. In *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems: 7th international conference, cPAIOR 2010, bologna, Italy, June 14-18, 2010. proceedings 7* (pp. 313–317). Springer.

Bianco, L., & Caramia, M. (2011). Minimizing the completion time of a project under resource constraints and feeding precedence relations: a Lagrangian relaxation based lower bound. *4OR. A Quarterly Journal of Operations Research*, *9*(4), 371–389.

Blaise, L., Artigues, C., & Benoist, T. (2020). Solution repair by inequality network propagation in LocalSolver. In *Parallel problem solving from nature–PPSN XVI: 16th international conference, PPSN 2020, leiden, the netherlands, September 5-9, 2020, proceedings, part i 16* (pp. 332–345). Springer.

Blazewicz, J., & Ecker, K. (1983). A linear time algorithm for restricted bin packing and scheduling problems. *Operations Research Letters*, *2*(2), 80–83.

Blazewicz, J., & Kubiak, W. (1989). Scheduling independent fixed-type tasks. In R. Slowinski, & J. Weglarz (Eds.), *Advances in project scheduling* (pp. 225–236). Elsevier.

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

Boctor, F. F. (1996). An adaptation of the simulated annealing algorithm for solving resource-constrained project scheduling problems. *International Journal of Production Research*, *34*, 2335–2351.

Bofill, M., Coll, J., Suy, J., & Villaret, M. (2020). SMT encodings for resource-constrained project scheduling problems. *Computers & Industrial Engineering*, *149*, Article 106777.

Bonifas, N. (2017). *Geometric and dual approaches to cumulative scheduling* (Ph.D. thesis), Université Paris-Saclay.

Boussemart, F., Hemery, F., Lecoutre, C., & Sais, L. (2004). Boosting systematic search by weighting constraints. *16*, In *ECAI* (p. 146).

Bredael, D., & Vanhoucke, M. (2023). Multi-project scheduling: A benchmark analysis of metaheuristic algorithms on various optimisation criteria and due dates. *European Journal of Operational Research*, *308*(1), 54–75.

Browning, T. R., & Yassine, A. A. (2010). Resource-constrained multi-project scheduling: Priority rule performance revisited. *International Journal of Production Economics*, *126*(2), 212–228.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., & Pesch, E. (1999). Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, *112*(1), 3–41.

Brucker, P., & Knust, S. (2000a). A linear programming and constraint propagation-based lower bound for the RCPSP. *European Journal of Operational Research*, *127*(2), 355–362.

Brucker, P., & Knust, S. (2000b). Resource-constrained project scheduling and timetabling. In *International conference on the practice and theory of automated timetabling* (pp. 277–293). Springer.

Brucker, P., & Knust, S. (2003). Lower bounds for resource-constrained project scheduling problems. *European Journal of Operational Research*, *149*(2), 302–313.

Brucker, P., & Knust, S. (2012). *Complex scheduling*. Springer Verlag.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998a). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *107*(2), 272–288.

Brucker, P., Knust, S., Schoo, A., & Thiele, O. (1998b). A branch and bound algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *107*, 272–288.

Brucker, P., & Krämer, A. (1996). Polynomial algorithms for resource-constrained and multiprocessor task scheduling problems. *European Journal of Operational Research*, *90*(2), 214–226.

Carlier, J., Jouglet, A., & Sahli, A. (2023). Algorithms to compute the energetic lower bounds of the cumulative scheduling problem. *Annals of Operations Research*, 1–31.

Carlier, J., & Latapie, B. (1991). Une méthode arborescente pour résoudre les problèmes cumulatifs. *RAIRO-Operations Research*, *25*(3), 311–340.

Carlier, J., & Néron, E. (2000). A new LP-based lower bound for the cumulative scheduling problem. *European Journal of Operational Research*, *127*(2), 363–382.

Carlier, J., & Néron, E. (2003). On linear lower bounds for the resource constrained project scheduling problem. *European Journal of Operational Research*, *149*(2), 314–324.

Carlier, J., & Néron, E. (2007). Computing redundant resources for the resource constrained project scheduling problem. *European Journal of Operational Research*, *176*(3), 1452–1463.

Carlier, J., & Pinson, É. (1989). An algorithm for solving the job-shop problem. *Management Science*, *35*(2), 164–176.

Carlier, J., & Pinson, E. (2004). Jackson's pseudo-preemptive schedule and cumulative scheduling problems. *Discrete Applied Mathematics*, *145*(1), 80–94.

Carlier, J., Pinson, E., Sahli, A., & Jouglet, A. (2020). An $O(n^2)$ algorithm for time-bound adjustments for the cumulative scheduling problem. *European Journal of Operational Research*, *286*(2), 468–476.

Carlier, J., Sahli, A., Jouglet, A., & Pinson, E. (2022). A faster checker of the energetic reasoning for the cumulative scheduling problem. *International Journal of Production Research*, *60*(11), 3419–3434.

Caseau, Y., & Laburthe, F. (1996). Cumulative scheduling with task intervals. *96*, In *JICSLP* (pp. 369–383). Citeseer.

Cavalcante, C., de Souza, C. C., Savelsbergh, M., Wang, Y., & Wolsey, L. (2001). Scheduling projects with labor constraints. *Discrete Applied Mathematics*, *112*(1–3), 27–52.

Christofides, N., Alvarez-Valdés, R., & Tamarit, J. M. (1987). Project scheduling with resource constraints: A branch and bound approach. *European Journal of Operational Research*, *29*(3), 262–273.

Chtourou, H., & Haouari, M. (2008). A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling. *Computers & Industrial Engineering*, *55*(1), 183–194.

Chu, G., Stuckey, P. J., Schutt, A., Ehlers, T., Gange, G., & Francis, K. (2018). Chuffed, a lazy clause generation solver. https://github.com/chuffed/chuffed.

Coelho, J., & Vanhoucke, M. (2018). An exact composite lower bound strategy for the resource-constrained project scheduling problem. *Computers and Operations Research*, *93*, 135–150.

Coelho, J., & Vanhoucke, M. (2020). Going to the core of hard resource-constrained project scheduling instances. *Computers and Operations Research*, *121*, Article 104976.

Coelho, J., & Vanhoucke, M. (2023). New resource-constrained project scheduling instances for testing (meta-)heuristic scheduling algorithms. *Computers and Operations Research*, *153*, Article 106165.

Coffman, E. G., Csirik, J., Galambos, G., Martello, S., Vigo, D., et al. (2013). Bin packing approximation algorithms: Survey and classification. In *Handbook of combinatorial optimization* (pp. 455–531). Springer.

Colin, J., & Vanhoucke, M. (2016). Empirical perspective on activity durations for project management simulation studies. *Journal of Construction Engineering and Management*, *142*(1), Article 04015047.

Dauzère-Pérès, S., & Lasserre, J. (1995). A new mixed-integer formulation of the flow-shop sequencing problem. In *2nd workshop on models and algorithms for planning and scheduling problems, wernigerode, allemagne*.

Davis, E. W. (1966). Resource allocation in project network models-a survey. *Journal of Industrial Engineering*, *17*(4), 177.

De Reyck, B., & Herroelen, W. (1995). Assembly line balancing by resource-constrained proect scheduling techniques: a critical approach. *DTEW Research Report 09505*, 1–32.

de Souza, C., & Wolsey, L. (1997). *Scheduling projects with labour constraints*: *Technical Report*, (IC-97-22), UNICAMP.

Debels, D., Reyck, B. D., Leus, R., & Vanhoucke, M. (2006). A hybrid scatter search / electromagnetism meta-heuristic for project scheduling. *European Journal of Operational Research*, *169*(2), 638–653.

Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project scheduling problems. *Operations Research*, *55*, 457–469.

Demassey, S. (2013). Mathematical programming formulations and lower bounds. In C. Artigues, S. Demassey, & E. Neron (Eds.), *Resource-constrained project scheduling: models, algorithms, extensions and applications* (pp. 49–62). John Wiley & Sons.

Demassey, S., Artigues, C., Baptiste, P., & Michelon, P. (2004). Lagrangean relaxation-based lower bounds for the RCPSP. In *Proceedings of PMS 2004* (pp. 76–79).

Demassey, S., Artigues, C., & Michelon, P. (2005). Constraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem. *INFORMS Journal on Computing*, *17*(1), 52–65.

Demeulemeester, E., & Heerroelen, W. (2002). Stochastic project scheduling. In F. S. Hillier (Ed.), *Project scheduling: a research handbook* (pp. 535–591). Boston, MA: Springer US, http://dx.doi.org/10.1007/0-306-48142-1_9.

Demeulemeester, E., & Heerroelen, W. (2013). Proactive-reactive project scheduling. In C. Artigues, S. Demassey, & E. Neron (Eds.), *Resource-constrained project scheduling: models, algorithms, extensions and applications* (pp. 203–211). John Wiley & Sons.

Demeulemeester, E., & Herroelen, W. (1992). A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, *38*, 1803–1818.

Demeulemeester, E., & Herroelen, W. (1997a). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, *43*, 1485–1492.

Demeulemeester, E. L., & Herroelen, W. S. (1997b). New benchmark results for the resource-constrained project scheduling problem. *Management Science*, *43*(11), 1485–1492.

Demeulemeester, E. L., & Herroelen, W. S. (2006). *vol. 49*, *Project scheduling: a research handbook*. Springer Science & Business Media.

Demeulemeester, E., Herroelen, W., et al. (2010). Robust project scheduling. *Foundations and Trends® in Technology, Information and Operations Management*, *3*(3–4), 201–376.

Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). RanGen: A random network generator for activity-on-the-node networks. *Journal of Scheduling*, *6*, 17–38.

Derrien, A., & Petit, T. (2014). A new characterization of relevant intervals for energetic reasoning. In *Principles and practice of constraint programming: 20th international conference, CP 2014, lyon, France, September 8-12, 2014. proceedings 20* (pp. 289–297). Springer.

Dike, S. H. (1964). Project scheduling with resource constraints. *IEEE Transactions on Engineering Management*, *EM-11*(4), 155–157.

Dorndorf, U. (2002). *Project scheduling with time windows: from theory to applications*. Springer Science & Business Media.

Dorndorf, U., Drexl, A., Nikulin, Y., & Pesch, E. (2007). Flight gate scheduling: State-of-the-art and recent developments. *Omega*, *35*(3), 326–334.

Dorndorf, U., Pesch, E., & Phan-Huy, T. (2000a). A branch-and-bound algorithm for the resource-constrained project scheduling problem. *Mathematical Methods of Operations Research*, *52*(3), 413–439.

Dorndorf, U., Pesch, E., & Phan-Huy, T. (2000b). Constraint propagation techniques for the disjunctive scheduling problem. *Artificial Intelligence*, *122*(1–2), 189–240.

Dorndorf, U., Pesch, E., & Phan-Huy, T. (2000c). A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, *46*(10), 1365–1384.

Downey, R. G., & Fellows, M. R. (2012). *Parameterized complexity*. Springer Science & Business Media.

Dumic, M., & Jakobovic, D. (2021). Ensembles of priority rules for resource constrained project scheduling problem. *Applied Soft Computing*, *110*, Article 107606.

Dürr, C. (2023). The scheduling zoo project. GitHub repository, https://github.com/xtof-durr/schedulingzoo/wiki/The-Scheduling-Zoo-project.

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, *230*(3), 449–463.

Elmaghraby, S. E. (1977). *Activity networks: Project planning and control by network models*. Wiley, New York.

Elmaghraby, S., & Herroelen, W. (1980). On the measurement of complexity in activity networks. *European Journal of Operational Research*, *5*, 223–234.

Erschler, J. (1976). *Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement* (Ph.D. thesis), (Thèse de docteur d'état) Université Paul Sabatier, Toulouse.

Erschler, J., Fontan, G., & Roubellat, F. (1979). Potentiels sur un graphe non conjonctif et analyse d'un problème d'ordonnancement à moyens limités. *RAIRO-Operations Research*, *13*(4), 363–378.

Even, C., Schutt, A., & Van Hentenryck, P. (2015). A constraint programming approach for non-preemptive evacuation scheduling. In *International conference on principles and practice of constraint programming* (pp. 574–591). Springer.

Fahimi, H., Ouellet, Y., & Quimper, C.-G. (2018). Linear-time filtering algorithms for the disjunctive constraint and a quadratic filtering algorithm for the cumulative not-first not-last. *Constraints*, *23*, 272–293.

Fahimi, H., & Quimper, C.-G. (2014). Linear-time filtering algorithms for the disjunctive constraint. *28*, In *Proceedings of the AAAI conference on artificial intelligence*.

Feydy, T., & Stuckey, P. J. (2009). Lazy clause generation reengineered. In *International conference on principles and practice of constraint programming* (pp. 352–366). Springer.

Fortemps, P., & Hapke, M. (1997). On the disjunctive graph for project scheduling. *Foundations of Computing and Decision Sciences*, *22*, 195–209.

Fu, F. (2014). Integrated scheduling and batch ordering for construction project. *Applied Mathematical Modelling*, *38*(2), 784–797.

Fündeling, C.-U., & Trautmann, N. (2010). A priority-rule method for project scheduling with work-content constraints. *European Journal of Operational Research*, *203*(3), 568–574.

Gafarov, E. R., Lazarev, A. A., & Werner, F. (2014). Approximability results for the resource-constrained project scheduling problem with a single type of resources. *Annals of Operations Research*, *213*(1), 115–130.

Ganian, R., Hamm, T., & Mescoff, G. (2020). The complexity landscape of resource-constrained scheduling. In C. Bessiere (Ed.), *Proceedings of the twenty-ninth international joint conference on artificial intelligence, IJCAI-20* (pp. 1741–1747).

Garey, M. R., & Graham, R. L. (1975). Bounds for multiprocessor scheduling with resource constraints. *SIAM Journal on Computing*, *4*(2), 187–200.

Garey, M. R., & Johnson, D. S. (1975). Complexity results for multiprocessor scheduling under resource constraints. *SIAM Journal on Computing*, *4*(4), 397–411.

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. Freeman.

Gay, S., Hartert, R., & Schaus, P. (2015). Simple and scalable time-table filtering for the cumulative constraint. In *Principles and practice of constraint programming: 21st international conference, CP 2015, cork, Ireland, August 31–September 4, 2015, proceedings 21* (pp. 149–157). Springer.

Gingras, V., & Quimper, C.-G. (2016). Generalizing the edge-finder rule for the cumulative constraint. In *IJCAI* (pp. 3103–3109).

Goldratt, E. M. (2017). *Critical chain: A business novel*. Routledge.

Gómez Sánchez, M., Lalla-Ruiz, E., Gil, A. F., Castro, C., & Voß, S. (2023). Resource-constrained multi-project scheduling problem: A survey. *European Journal of Operational Research*, *309*(3), 958–976.

Goyal, D. (1976). *Scheduling equal execution time tasks under unit resource restriction* (Ph.D. thesis), Washington State University.

Graham, R. L., Lawler, E. L., Lenstra, J. K., & Rinnooy Kan, A. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *vol. 5*, In *Annals of discrete mathematics* (pp. 287–326). Elsevier.

Grimes, D., & Hebrard, E. (2015). Solving variants of the job shop scheduling problem through conflict-directed search. *INFORMS Journal on Computing*, *27*(2), 268–284.

Guo, W., Vanhoucke, M., & Coelho, J. (2023). A prediction model for ranking branch-and-bound procedures for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *306*, 579–595.

Guo, W., Vanhoucke, M., Coelho, J., & Luo, J. (2021). Automatic detection of the best performing priority rule for the resource-constrained project scheduling problem. *Expert Systems with Applications*, *167*, Article 114116.

Haouari, M., Koolu, A., Néron, E., & Carlier, J. (2014). A preemptive bound for the resource constrained project scheduling problem. *Journal of Scheduling*, *17*, 237–248.

Hardin, J. R., Nemhauser, G. L., & Savelsbergh, M. W. (2008). Strong valid inequalities for the resource-constrained scheduling problem with uniform resource requirements. *Discrete Optimization*, *5*(1), 19–35.

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

Hartmann, S. (2002). A self-adapting genetic algorithm for project scheduling under resource constraints. *Naval Research Logistics*, *49*, 433–448.

Hartmann, S. (2013). Project scheduling with resource capacities and requests varying with time: A case study. *Flexible Services and Manufacturing Journal*, *25*(1), 74–93.

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. (2022). An updated survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *297*(1), 1–14.

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

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *127*(2), 394–407.

Hebrard, E. (2017). Resource constraints in scheduling. Association for Constraint Programming (ACP) summer school, https://homepages.laas.fr/ehebrard/papers/lecture2017.pdf.

Heinz, V., Hanzálek, Z., & Vilím, P. (2024). Reinforcement learning for search tree size minimization in constraint programming: New results on scheduling benchmarks. Available At SSRN 4938242.

Herroelen, W. S. (1972). Resource-constrained project scheduling—the state of the art. *Journal of the Operational Research Society*, *23*(3), 261–275.

Herroelen, W., & De Reyck, B. (1999). Phase transitions in project scheduling. *Journal of the Operational Research Society*, *50*, 148–156.

Herroelen, W., De Reyck, B., & Demeulemeester, E. (1998). Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research*, *25*(4), 279–302.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (1999). A classification scheme for project scheduling problems. In J. Weglarz (Ed.), *Project scheduling - recent models, algorithms and applications* (pp. 1–26). Dortrecht, Kluwer Academic Publishers.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (2001). A note on the paper "resource-constrained project scheduling: Notation, classification, models and methods" by brucker et al.. *European Journal of Operational Research*, *128*(3), 679–688.

Herroelen, W., & Leus, R. (2001). On the merits and pitfalls of critical chain scheduling. *Journal of Operations Management*, *19*(5), 559–577.

Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*(2), 289–306.

Hill, A., Lalla-Ruiz, E., Voß, S., & Goycoolea, M. (2019). A multi-mode resource-constrained project scheduling reformulation for the waterway ship scheduling problem. *Journal of Scheduling*, *22*, 173–182.

van Hoeve, W.-J., & Katriel, I. (2006). Global constraints. In F. Rossi, P. van Beek, & T. Walsh (Eds.), *Handbook of Constraint Programming*, 205.

Homberger, J. (2007). A multi-agent system for the decentralized resource-constrained multi-project scheduling problem. *International Transactions in Operational Research*, *14*(6), 565–589.

Horbach, A. (2010). A boolean satisfiability approach to the resource-constrained project scheduling problem. *Annals of Operations Research*, *181*, 89–107.

Icmeli, O., Selcuk Erenguc, S., & Zappe, C. J. (1993). Project scheduling problems: a survey. *International Journal of Operations & Production Management*, *13*(11), 80–91.

Kameugne, R., Betmbe, S. F., Noulamo, T., & Djamegni, C. T. (2023). Horizontally elastic edge finder rule for cumulative constraint based on slack and density. In *29th international conference on principles and practice of constraint programming (CP 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Kameugne, R., Betmbe Fetgo, S., Gingras, V., Ouellet, Y., & Quimper, C.-G. (2018). Horizontally elastic not-first/not-last filtering algorithm for cumulative resource constraint. In *International conference on the integration of constraint programming, artificial intelligence, and operations research* (pp. 316–332). Springer.

Kameugne, R., Fetgo, S. B., Fotso, L. P., et al. (2013). Energetic extended edge finding filtering algorithm for cumulative resource constraints. *American Journal of Operations Research*, *3*(06), 589.

Kameugne, R., & Fotso, L. P. (2010). A complete filtering algorithm for cumulative not-first/not-last rule in O (n 2| H| log n). *Proceeding of CSCLP*, 31–42.

Kameugne, R., & Fotso, L. P. (2013). A cumulative not-first/not-last filtering algorithm in O (n 2 log (n)). *Indian Journal of Pure and Applied Mathematics*, *44*, 95–115.

Kameugne, R., Fotso, L. P., Scott, J., & Ngo-Kateu, Y. (2014). A quadratic edge-finding filtering algorithm for cumulative resource constraints. *Constraints*, *19*, 243–269.

Kelley, J. (1963). The critical path method: resources planning and scheduling. In J. F. Muth, & G. L. Thompson (Eds.), *Industrial scheduling*. Englewood Cliffs: Prentice-Hall.

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).

Khoshjahan, Y., Najafi, A. A., & Afshar-Nadjafi, B. (2013). Resource constrained project scheduling problem with discounted earliness–tardiness penalties: Mathematical modeling and solving procedure. *Computers & Industrial Engineering*, *66*(2), 293–300.

Klein, R., & Scholl, A. (1999). Computing lower bounds by destructive improvement: An application to resource-constrained project scheduling. *European Journal of Operational Research*, *112*(2), 322–346.

Knust, S. (2010). Scheduling non-professional table-tennis leagues. *European Journal of Operational Research*, *200*(2), 358–367.

Knust, S. (2015). Lower bounds on the minimum project duration. *Handbook on Project Management and Scheduling Vol. 1*, 43–55.

Kobylanski, P., & Kuchta, D. (2007). A note on the paper by M.A. Al-Fawzan and M. Haouari about a bi-objective problem for robust resource-constrained project scheduling. *International Journal of Production Economics*, *107*, 496–501.

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. (2015). Shifts, types, and generation schemes for project schedules. *Handbook on Project Management and Scheduling Vol. 1*, 3–16.

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

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

Kolisch, R., & Padman, R. (2001). An integrated survey of deterministic project scheduling. *Omega*, *29*(3), 249–272.

Kolisch, R., & Sprecher, A. (1996). PSPLIB - a project scheduling problem library. *European Journal of Operational Research*, *96*, 205–216.

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*(1), 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.

Koné, O., Artigues, C., Lopez, P., & Mongeau, M. (2011). Event-based MILP models for resource-constrained project scheduling problems. *Computers & Operations Research*, *38*(1), 3–13.

Kovalyov, M. Y., & Shafransky, Y. M. (1998). Uniform machine scheduling of unit-time jobs subject to resource constraints. *Discrete Applied Mathematics*, *84*(1–3), 253–257.

Krause, K. L., Shen, V. Y., & Schwetman, H. D. (1975). Analysis of several task-scheduling algorithms for a model of multiprogramming computer systems. *Journal of the ACM*, *22*(4), 522–550.

Krüger, D., & Scholl, A. (2010). Managing and modelling general resource transfers in (multi-) project scheduling. *OR Spectrum*, *32*, 369–394.

Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, *143*(2), 151–188.

Laborie, P. (2005). Complete MCS-based search: Application to resource constrained project scheduling. In *IJCAI* (pp. 181–186). Citeseer.

Laborie, P. (2023). Bornes rapides pour l'ordonnancement dans LocalSolver. In *24ème congrès de la Société Française de Recherche Opérationnelle et d'Aide à la Décision*.

Lahrichi, A. (1982). Ordonnancements. La notion de "parties obligatoires" et son application aux problèmes cumulatifs. *RAIRO-Operations Research*, *16*(3), 241–262.

Lasserre, J. B., & Queyranne, M. (1992). Generic scheduling polyhedra and a new mixed-integer formulation for single-machine scheduling. In d. G. C. E. Balas a, & R. Kannan (Eds.), *Integer programming and combinatorial optimization – proceedings of the 2nd international IPCO conference* (pp. 136–149).

Le Pape, C. (1994). Implementation of resource constraints in ILOG schedule: A library for the development of constraint-based scheduling systems. *Intelligent Systems Engineering*, *3*(2), 55–66.

Le Pape, C. (1995). Three mechanisms for managing resource constraints in a library for constraint-based scheduling. *1*, In *Proceedings 1995 INRIA/IEEE symposium on emerging technologies and factory automation. eTFA'95* (pp. 281–289). IEEE.

Le Pape, C., Couronne, P., Vergamini, D., & Gosselin, V. (1994). Time-versus-capacity compromises in projectscheduling. In *Proceedings of the thirteenth workshop of the U.k. planning special interest group*.

Lenstra, J. K., & Rinnooy Kan, A. (1978). Complexity of scheduling under precedence constraints. *Operations Research*, *26*(1), 22–35.

Letort, A., Beldiceanu, N., & Carlsson, M. (2012). A scalable sweep algorithm for the cumulative constraint. In *International conference on principles and practice of constraint programming* (pp. 439–454). Springer.

Leyman, P., & Vanhoucke, M. (2017). Capital- and resource-constrained project scheduling with net present value optimization. *European Journal of Operational Research*, *256*(3), 757–776.

Liess, O., & Michelon, P. (2008). A constraint programming approach for the resource-constrained project scheduling problem. *Annals of Operations Research*, *157*(1), 25–36.

Lopez, P. (1991). *Approche énergétique pour l'ordonnancement de tâches sous contraintes de temps et de ressources* (Ph.D. thesis), Université Paul Sabatier - Toulouse III.

Lozano, R. C., & Schulte, C. (2019). Survey on combinatorial register allocation and instruction scheduling. *ACM Computing Surveys*, *52*(3), 1–50.

Luo, J., Vanhoucke, M., Coelho, J., & Guo, W. (2022). An efficient genetic programming approach to design priority rules for resource-constrained project scheduling problem. *Expert Systems with Applications*, *198*, Article 116753.

Malcolm, D. G., Roseboom, J. H., Clark, C. E., & Fazar, W. (1959). Application of a technique for research and development program evaluation. *Operations Research*, *7*(5), 646–669.

Mastor, A. (1970). An experimental and comparative evaluation of production line balancing techniques. *Management Science*, *16*, 728–746.

Mercier, L., & Van Hentenryck, P. (2008). Edge finding for cumulative scheduling. *INFORMS Journal on Computing*, *20*(1), 143–153.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., & Bianco, L. (1998). An exact algorithm for the resource constrained project scheduling problem based on a new mathematical formulation. *Management Science*, *44*, 714–729.

Möhring, R. H., Schulz, A. S., Stork, F., & Uetz, M. (2003). Solving project scheduling problems by minimum cut computations. *Management Science*, *49*(3), 330–350.

Morin, P.-A., Artigues, C., Haït, A., Kis, T., & Spieksma, F. C. (2022). A project scheduling problem with periodically aggregated resource-constraints. *Computers & Operations Research*, *141*, Article 105688.

Morrison, D. R., Jacobson, S. H., Sauppe, J. J., & Sewell, E. C. (2016). Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, *19*, 79–102. http://dx.doi.org/10.1016/j.disopt.2016.01.005, URL https://www.sciencedirect.com/science/article/pii/S1572528616000062.

Moukrim, A., Quilliot, A., & Toussaint, H. (2015). An effective branch-and-price algorithm for the preemptive resource constrained project scheduling problem based on minimal interval order enumeration. *European Journal of Operational Research*, *244*(2), 360–368.

Naber, A., & Kolisch, R. (2014). MIP models for resource-constrained project scheduling with flexible resource profiles. *European Journal of Operational Research*, *239*(2), 335–348.

Narbaev, T., Hazir, Ö., Khamitova, B., & Talgat, S. (2024). A machine learning study to improve the reliability of project cost estimates. *International Journal of Production Research*, *62*, 4372–4388.

Nazareth, T., Verma, S., Bhattacharya, S., & Bagchi, A. (1999). The multiple resource constrained project scheduling problem: A breadth-first approach. *European Journal of Operational Research*, *112*(2), 347–366.

Néron, E. (2008). Resource and precedence constraint relaxation. *Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications*, 37–48.

Neron, E., Artigues, C., Baptiste, P., Carlier, J., Damay, J., Demassey, S., & Laborie, P. (2006). Lower bounds for resource constrained project scheduling problem: Recent advances. *Perspectives in Modern Project Scheduling*, 167–204.

Neumann, K., Nübel, H., & Schwindt, C. (2000). Active and stable project scheduling. *Mathematical Methods of Operations Research*, *52*, 441–465.

Neumann, K., & Schwindt, C. (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research*, *56*, 513–533.

Neumann, K., Schwindt, C., & Zimmermann, J. (2002). *vol. 508*, *Project scheduling with time windows and scarce resources: temporal and resource-constrained project scheduling with regular and nonregular objective functions*. Springer.

Nikoofal Sahl Abadi, N., Bagheri, M., & Assadi, M. (2018). Multiobjective model for solving resource-leveling problem with discounted cash flows. *International Transactions in Operational Research*, *25*(6), 2009–2030.

Nuijten, W. P. M. (1994). *Time and resource constrained scheduling: a constraint satisfaction approach* (Ph.D. thesis), Eindhoven University of Technology.

Ohrimenko, O., Stuckey, P. J., & Codish, M. (2009). Propagation via lazy clause generation. *Constraints*, *14*, 357–391.

Ouellet, P., & Quimper, C.-G. (2013). Time-table extended-edge-finding for the cumulative constraint. In *Principles and practice of constraint programming: 19th international conference, CP 2013, uppsala, Sweden, September 16-20, 2013. proceedings 19* (pp. 562–577). Springer.

Ouellet, Y., & Quimper, C.-G. (2018). A $O(n \log^2 n)$ checker and $O(n^2 \log n)$ filtering algorithm for the energetic reasoning. In *International conference on the integration of constraint programming, artificial intelligence, and operations research* (pp. 477–494). Springer.

Özdamar, L., & Ulusoy, G. (1995). A survey on the resource-constrained project scheduling problem. *IIE Transactions*, *27*(5), 574–586.

Palpant, M., Artigues, C., & Michelon, P. (2004). LSSPER: Solving the resource-constrained project scheduling problem with large neighbourhood search. *Annals of Operations Research*, *131*, 237–257.

Paraskevopoulos, D., Tarantilis, C., & Ioannou, G. (2012). Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications*, *39*(4), 3983–3994.

Pascoe, T. (1966). Allocation of resources - CPM. *Revue FranÇaise de Recherche Opérationnelle*, *38*, 31–38.

Patterson, J. (1976). Project scheduling: The effects of problem structure on heuristic scheduling. *Naval Research Logistics*, *23*, 95–123.

Patterson, J. (1984). A comparison of exact approaches for solving the multiple constrained resource project scheduling problem. *Management Science*, *30*, 854–867.

Patterson, W. (1974). A horizon-varying, zero-one approach to project scheduling. *Management Science*, *20*, 990–998.

Pellerin, R., Perrier, N., & Berthaut, F. (2020). A survey of hybrid metaheuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *280*(2), 395–416.

Perregaard, M. (1995). Branch and bound methods for the multi-processor job-shop and flow-shop scheduling problems. *Master's Thesis, Datalogisk Institute Kubenhavns Universitet*.

Perron, L., & Didier, F. (2023). CP-SAT. https://developers.google.com/optimization/cp/cp_solver/, Google.

Polo-Mejía, O., Artigues, C., Lopez, P., Mönch, L., & Basini, V. (2023). Heuristic and metaheuristic methods for the multi-skill project scheduling problem with partial preemption. *International Transactions in Operational Research*, *30*(2), 858–891.

Ponz-Tienda, J., Salcedo-Bernal, A., Pellicer, E., & Benlloch-Marco, J. (2017). Improved adaptive harmony search algorithm for the resource leveling problem with minimal lags. *Automation in Construction*, *77*, 82–92.

Pritsker, A., & Watters, L. (1968). *A zero-one programming approach to schedul- ing with limited resources*: *Technical Report*, (RM-5561-P), The RAND Corporation.

Proon, S., & Jin, M. (2011). A genetic algorithm with neighborhood search for the resource-constrained project scheduling problem. *Naval Research Logistics*, *58*(2), 73–82.

Prud'homme, C., & Fages, J.-G. (2022). Choco-solver. *Journal of Open Source Software*, *7*(78), 4708.

Queyranne, M., & Schulz, A. S. (1994). *Polyhedral approaches to machine scheduling*. Citeseer.

Quintanilla, S., Lino, P., Pérez, Á., Ballestín, F., & Valls, V. (2015). Integer preemption problems. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on project management and scheduling vol. 1* (pp. 231–250). Springer.

Rieck, J., Zimmermann, J., & Gather, T. (2012). Mixed-integer linear programming for resource leveling problems. *European Journal of Operational Research*, *221*(1), 27–37.

Riise, A., Mannino, C., & Lamorgese, L. (2016). Recursive logic-based benders' decomposition for multi-mode outpatient scheduling. *European Journal of Operational Research*, *255*(3), 719–728.

Rossi, F., Van Beek, P., & Walsh, T. (2006). *Handbook of constraint programming*. Elsevier.

Roux, W. (1998). Multi-resource shop scheduling with resource flexibility. *European Journal of Operational Research*, *107*(2), 289–305.

Roy, B., & Dibon, M. (1966). L'ordonnancement par la méthode des potentiels–le programme concord. *Automatisme*, *2*, 1–11.

Sankaran, J. K., Bricker, D. L., & Juang, S.-H. (1999). A strong fractional cutting-plane algorithm for resource-constrained project scheduling. *International Journal of Industrial Engineering*, *6*, 99–111.

Scales, J. (2020). A design science research approach to closing the gap between the research and practice of project scheduling. *Systems Research and Behavioral Science*, *37*(5), 804–812.

Schäffter, M. W. (1997). Scheduling with forbidden sets. *Discrete Applied Mathematics*, *72*(1–2), 155–166.

Schutt, A., Feydy, T., & Stuckey, P. J. (2013). Explaining time-table-edge-finding propagation for the cumulative resource constraint. In *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems: 10th international conference, cPAIOR 2013, yorktown heights, NY, USA, May 18-22, 2013. proceedings 10* (pp. 234–250). Springer.

Schutt, A., Feydy, T., Stuckey, P. J., & Wallace, M. G. (2009). Why cumulative decomposition is not as bad as it sounds. In *Principles and practice of constraint programming-CP 2009: 15th international conference, CP 2009 lisbon, Portugal, September 20-24, 2009 proceedings 15* (pp. 746–761). Springer.

Schutt, A., Feydy, T., Stuckey, P. J., & Wallace, M. G. (2011). Explaining the cumulative propagator. *Constraints*, *16*, 250–282.

Schutt, A., Feydy, T., Stuckey, P. J., & Wallace, M. G. (2015). A satisfiability solving approach. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on project management and scheduling vol. 1* (pp. 135–160). Springer.

Schutt, A., Wolf, A., & Schrader, G. (2005). Not-first and not-last detection for cumulative scheduling in. In *International conference on applications of declarative programming and knowledge management* (pp. 66–80). Springer.

Schwindt, C. (1995). A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags. *WIOR-Report-449. Institut Für Wirtschaftstheorie Und Operations Research, University of Karlsruhe*.

Schwindt, C. (1998). *Verfahren zur lösung des ressourcenbeschränkten projektdauermin-imierungsproblems mit planungsabhängigen zeitfenstern* (Ph.D. thesis), Fakultät für wirtschaftswissenschaften der Universität Fridericiana zu Karlsruhe.

Schwindt, C., & Trautmann, N. (2000). Batch scheduling in process industries: an application of resource–constrained project scheduling. *OR-Spektrum*, *22*, 501–524.

Schwindt, C., Zimmermann, J., et al. (2015). *Handbook on project management and scheduling vol. 1*. Springer.

Servranckx, T., & Vanhoucke, M. (2019). A tabu search procedure for the resource-constrained project scheduling problem with alternative subgraphs. *European Journal of Operational Research*, *273*(3), 841–860.

Shariatmadari, M., Nahavandi, N., Zegordi, S. H., & Sobhiyah, M. H. (2017). Integrated resource management for simultaneous project selection and scheduling. *Computers & Industrial Engineering*, *109*, 39–47.

Shuvo, O., Golder, S., & Islam, M. R. (2023). A hybrid metaheuristic method for solving resource constrained project scheduling problem. *Evolutionary Intelligence*, *16*(2), 519–537.

Simonin, G., Artigues, C., Hebrard, E., & Lopez, P. (2015). Scheduling scientific experiments for comet exploration. *Constraints*, *20*, 77–99.

Sittel, P., Kumm, M., Oppermann, J., Möller, K., Zipf, P., & Koch, A. (2018). ILP-based modulo scheduling and binding for register minimization. In *2018 28th international conference on field programmable logic and applications* (pp. 265–2656). IEEE.

Slowinski, R. (1980). Two approaches to problems of resource allocation among project activities – a comparative study. *Journal of Operational Research Society*, *8*, 711–723.

Slowinski, R. (1981). Multiobjective network scheduling with efficient use of renewable and nonrenewable resources. *European Journal of Operational Research*, *7*, 265–273.

Slowinski, R., Soniewicki, B., & Weglarz, J. (1994). DSS for multiobjective project scheduling. *European Journal of Operational Research*, *79*(2), 220–229.

Snauwaert, J., & Vanhoucke, M. (2023). A classification and new benchmark instances for the multi-skilled resource-constrained project scheduling problem. *European Journal of Operational Research*, *307*(1), 1–19.

Sousa, J. (1989). *Time indexed formulations of non-preemptive single-machine scheduling problems* (Ph.D. thesis), Université Catholique de Louvain.

Sprecher, A. (2000). Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, *46*, 710–723.

Sprecher, A., Kolisch, R., & Drexl, A. (1995). Semi-active, active, and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *80*(1), 94–102.

Stinson, J., Davis, E., & Khumawala, B. (1978). Multiple resource-constrained scheduling using branch-and-bound. *IIE Transactions*, *10*, 252–259.

Talbot, F., & Patterson, J. (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, *24*, 1163–1174.

Tavares, L. (1999). *Advanced models for project management*. Kluwer Academic Publishers, Dordrecht, 1999.

Teichteil-Königsbuch, F., Povéda, G., González de Garibay Barba, G., Luchterhand, T., & Thiébaux, S. (2023). Fast and robust resource-constrained scheduling with graph neural networks. In *Proceedings of the 33rd international conference on automated planning and scheduling, ICAPS '23* (pp. 623–633). AAAI Press.

Tesch, A. (2016). A nearly exact propagation algorithm for energetic reasoning in. In *International conference on principles and practice of constraint programming* (pp. 493–519). Springer.

Tesch, A. (2018). Improving energetic propagations for cumulative scheduling. In *International conference on principles and practice of constraint programming* (pp. 629–645). Springer.

Tesch, A. (2020). A polyhedral study of event-based models for the resource-constrained project scheduling problem. *Journal of Scheduling*, *23*(2), 233–251.

Tormos, P., & Lova, A. (2001). A competitive heuristic solution technique for resource-constrained project scheduling. *Annals of Operations Research*, *102*, 65–81.

Trautmann, N., & Baumann, P. (2009). Resource-allocation capabilities of commercial project management software: an experimental analysis. In *2009 international conference on computers & industrial engineering* (pp. 1143–1148). IEEE.

Trietsch, D., Mazmanyan, L., Govergyan, L., & Baker, K. R. (2012). Modeling activity times by the parkinson distribution with a lognormal core: Theory and validation. *European Journal of Operational Research*, *216*, 386–396.

Uddin, S., Ong, S., Lu, H., & Matous, P. (2023). Integrating machine learning and network analytics to model project cost, time and quality performance. *Production Planning and Control*, *To Appear*.

Uetz, M. (2001). *Algorithms for deterministic and stochastic scheduling* (Ph.D. thesis), Technical University Berlin.

Ulusoy, G., & Hazır, Ö. (2021). Project scheduling under uncertainty. In *An introduction to project modeling and planning* (pp. 357–379). Cham: Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-61423-2_12.

Unsal-Altuncan, I., & Vanhoucke, M. (2024). A hybrid forecasting model to predict the duration and cost performance of projects with Bayesian networks. *European Journal of Operational Research*, *315*, 511–527.

Valls, V., Ballestin, F., & Quintanilla, M. S. (2005). Justification and RCPSP: A technique that pays. *European Journal of Operational Research*, *165*, 375–386.

Valls, V., Ballestin, F., & Quintanilla, S. (2008). A hybrid genetic algorithm for the resource-constrained project scheduling problem. *European Journal of Operational Research*, *185*(2), 495–508.

Van Bevern, R., Bredereck, R., Bulteau, L., Komusiewicz, C., Talmon, N., & Woeginger, G. J. (2016). Precedence-constrained scheduling problems parameterized by partial order width. In *International conference on discrete optimization and operations research* (pp. 105–120). Springer.

Van Eynde, R., & Vanhoucke, M. (2020). Resource-constrained multi-project scheduling: Benchmark datasets and decoupled scheduling. *Journal of Scheduling*, *23*, 301–325.

Van Eynde, R., & Vanhoucke, M. (2022a). New summary measures and datasets for the multi-project scheduling. *European Journal of Operational Research*, *299*, 853–868.

Van Eynde, R., & Vanhoucke, M. (2022b). A theoretical framework for instance complexity of the resource-constrained project scheduling problem. *Mathematics of Operations Research*, *47*, 3156–3183.

Van Eynde, R., Vanhoucke, M., & Coelho, J. (2024). On the summary measures for the resource-constrained project scheduling problem. *Annals of Operations Research*, *337*, 593–625.

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*(1), 62–72.

Van Peteghem, V., & Vanhoucke, M. (2015). Heuristic methods for the resource availability cost problem. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on project management and scheduling vol. 1* (pp. 339–359). Springer.

Vandevelde, A., Hoogeveen, H., Hurkens, C., & Lenstra, J. K. (2005). Lower bounds for the head-body-tail problem on parallel machines: a computational study of the multiprocessor flow shop. *INFORMS Journal on Computing*, *17*(3), 305–320.

Vanhoucke, M. (2012). *XVIII, Project Management with Dynamic Scheduling: Baseline Scheduling, Risk Analysis and Project Control*. Springer.

Vanhoucke, M. (2013). Project baseline scheduling: An overview of past experiences. *Journal of Modern Project Management*, *1*(2), 18–27.

Vanhoucke, M. (2023). *The illusion of control: Project data, computer algorithms and human intuition for project management and control*. Springer.

Vanhoucke, M. (2024). A quest for projects with scarce resources: Seeking schedule intelligence through project data discovery. In *Business guides on the go*, Springer, http://dx.doi.org/10.1007/978-3-031-71507-5.

Vanhoucke, M., & Batselier, J. (2019a). Fitting activity distributions using human partitioning and statistical calibration. *Computers & Industrial Engineering*, *129*, 126–135.

Vanhoucke, M., & Batselier, J. (2019b). A statistical method for estimating activity uncertainty parameters to improve project forecasting. *Entropy*, *21*, 952.

Vanhoucke, M., & Coelho, J. (2016). An approach using SAT solvers for the RCPSP with logical constraints. *European Journal of Operational Research*, *249*(2), 577–591.

Vanhoucke, M., & Coelho, J. (2018). A tool to test and validate algorithms for the resource-constrained project scheduling problem. *Computers & Industrial Engineering*, *118*, 251–265.

Vanhoucke, M., & Coelho, J. (2019). Resource-constrained project scheduling with activity splitting and setup times. *Computers & Operations Research*, *109*, 230–249.

Vanhoucke, M., & Coelho, J. (2021). An analysis of network and resource indicators for resource-constrained project scheduling problem instances. *Computers and Operations Research*, *132*, Article 105260.

Vanhoucke, M., & Coelho, J. (2024). A matheuristic for the resource- constrained project scheduling problem. *European Journal of Operational Research*, *319*, 711–725.

Vanhoucke, M., Coelho, J., Debels, D., Maenhout, B., & Tavares, L. (2008). An evaluation of the adequacy of project network generators with systematically sampled networks. *European Journal of Operational Research*, *187*, 511–524.

Vanhoucke, M., Demeulemeester, E. L., & Herroelen, W. S. (2001a). An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. *Annals of Operations Research*, *102*, 179–196.

Vanhoucke, M., Demeulemeester, E., & Herroelen, W. (2001b). On maximizing the net present value of a project under renewable resource constraints. *Management Science*, *47*, 1113–1121.

Vázquez, E. P., Calvo, M. P., & Ordóñez, P. M. (2015). Learning process on priority rules to solve the RCMPSP. *Journal of Intelligent Manufacturing*, *26*(1), 123–138.

Vilím, P. (2004). O (nlog n) filtering algorithms for unary resource constraint. In *Integration of AI and OR techniques in constraint programming for combinatorial optimization problems: first international conference, cPAIOR 2004, nice, France, April 20-22, 2004. proceedings 1* (pp. 335–347). Springer.

Vilım, P. (2007). *Global constraints in scheduling* (Ph.D. thesis), Charles University.

Vilím, P. (2009a). Edge finding filtering algorithm for discrete cumulative resources in. In *International conference on principles and practice of constraint programming* (pp. 802–816). Springer.

Vilím, P. (2009b). Max energy filtering algorithm for discrete cumulative resources. In *International conference on integration of constraint programming, artificial intelligence, and operations research* (pp. 294–308). Springer.

Vilím, P. (2011). Timetable edge finding filtering algorithm for discrete cumulative resources. In *International conference on AI and OR techniques in constriant programming for combinatorial optimization problems* (pp. 230–245). Springer.

Vilím, P., Laborie, P., & Shaw, P. (2015). Failure-directed search for constraint-based scheduling. In *Integration of AI and OR techniques in constraint programming: 12th international conference, CPAIOR 2015, Barcelona, Spain, May 18-22, 2015, proceedings 12* (pp. 437–453). Springer.

Wang, J., Hu, X., Demeulemeester, E., & Zhao, Y. (2021). A bi-objective robust resource allocation model for the RCPSP considering resource transfer costs. *International Journal of Production Research*, *59*(2), 367–387.

Watermeyer, K., & Zimmermann, J. (2023). A constructive branch-and-bound algorithm for the project duration problem with partially renewable resources and general temporal constraints. *Journal of Scheduling*, *26*(1), 95–111.

Wauters, M., & Vanhoucke, M. (2014). Support vector machine regression for project control forecasting. *Automation in Construction*, *47*, 92–106.

Wauters, M., & Vanhoucke, M. (2016). A comparative study of artificial intelligence methods for project duration forecasting. *Expert Systems with Applications*, *46*, 249–261.

Weglarz, J., Jozefowska, J., Mika, M., & Waligora, G. (2011). Project scheduling with finite or infinite number of activity processing modes – a survey. *European Journal of Operational Research*, *208*, 177–205.

Wiest, J. D. (1964). Some properties of schedules for large projects with limited resources. *Operations Research*, *12*(3), 395–418.

Wolf, A., & Schrader, G. (2005). Overload checking for the cumulative constraint and its application. In *International conference on applications of declarative programming and knowledge management* (pp. 88–101). Springer.

Wolsey, L. A. (1997). MIP modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research*, *99*(1), 154–165.

Zaloom, V. (1971). On the resource constrained project scheduling problem. *AIIE Transactions*, *3*(4), 302–305.

Zhao, X., Song, W., Li, Q., Shi, H., Kang, Z., & Zhang, C. (2022). A deep reinforcement learning approach for resource-constrained project scheduling. In *2022 IEEE symposium series on computational intelligence* (pp. 1226–1234). IEEE.