MISTA 2015

Exact and heuristic methods for trading-off makespan and stability in stochastic project scheduling

Simon Mountakis · Tomas Klos · Cees Witteveen · Bob Huisman

Abstract This paper addresses a problem of practical value in project scheduling: trading expected makespan for stability, under stochastic activity duration uncertainty. We present the formal statement of a problem that we name Proactive Stochastic RCPSP (PS-RCPSP). Assuming activity durations follow known probability distributions, PS-RCPSP asks to find a so-called earliest-start (es) policy and a proactive schedule that together minimize the weighted sum of expected project makespan and expected instability (deviation of the realized from the proactive schedule). Extending an existing MILP model for the well-known deterministic Resource-Constrained Project Scheduling Problem (RCPSP), we present a MILP model for PS-RCPSP, which allows us to find optimal (es-policy, proactive schedule) pairs. To deal with instances of practical size, we propose a Linear Programming (LP)-based and a Mixed-Integer LP (MILP)-based heuristic. Our LP-based heuristic optimizes the proactive schedule while keeping the es-policy part of the solution fixed. Our MILP-based heuristic aims to optimize the structure of the policy together with the proactive schedule. In contrast to existing state-of-art approaches such as CCP [21] and STC [31], our heuristics rely on optimizing the proactive schedule together with the scheduling policy. Experiments show that the LP-based heuristic is efficient and compares favorably with the state-of-art (i.e. achieves smaller expected makespan for a certain level of expected instability) when the aim is to achieve near-zero instability at the cost of higher makespan. The MILP-based heuristic seems more effective (albeit not as efficient) when the aim is to achieve low expected makespan at the cost of moderate or high instability.

Simon Mountakis
Delft University of Technology
E-mail: k.s.mountakis@tudelft.nl

Tomas Klos
Delft University of Technology
E-mail: t.b.klos@tudelft.nl

Cees Witteveen
Delft University of Technology
E-mail: c.witteveen@tudelft.nl

Bob Huisman
NedTrain
E-mail: b.huisman@nedtrain.nl

1 Introduction

Project scheduling literature mostly concentrates on scheduling subject to temporal and resource constraints. The schedule sought for is an assignment of start-times to activities, facilitating the efficient use of limited resources in order to minimize a lateness measure such as the project makespan. Finding a schedule usually invovles solving the Resource-Constrained Project Scheduling Problem (RCPSP) (see [1, 14] for comprehensive surveys). This problem has been shown to be NP-Hard [7] and finds several industrial applications (e.g. [8,6]). Associated literature includes numerous exact and (meta-)heuristic algorithms, able to find good schedules for large instances and diverse lateness measures (e.g. [28, 20, 18, 10]). Solving an RCPSP serves the purpose of preparing a feasible schedule, assuming a static deterministic project execution environment. In practice, however, this assumption is rarely valid. Activity durations used for preparing the schedule are mostly rough estimates, since most projects are subject to delays during execution and the final realized schedule is the result of subjecting the original schedule to modifications which make it consistent with the project constraints in the face of delays. Ad-hoc modifications lead to realized activity start-times that might differ from planned start-times, compromising project predictability and timeliness.

This paper addresses the issue of hedging against project uncertainty by preparing a schedule in combination with an execution strategy for coping with delays. In line with other works in *stochastic project scheduling* (see [17] for a comprehensive review) we assume activity durations are given as stochastic variables with known distributions and propose a new *proactive-reactive* scheduling method. First, we define the Proactive Stochastic (PS) RCPSP as an extension of RCPSP. The solution to PS-RCPSP is a *proactive schedule* and an *earliest-start (es-) policy* that together minimize the weighted sum of the realized schedule's expected makespan and its expected deviation from the proactive schedule. PS-RCPSP is, in fact, a generalization of the Stochastic RCPSP (S-RCPSP), which asks to find a *stochastic scheduling policy* instead of a schedule and various *classes* of policies have been proposed in the literature [25, 26, 17]. In general, a policy defines a mapping between activity duration realizations to realized schedules. S-RCPSP asks to find a policy that minimizes the expected project makespan, with only few exact and heuristic approaches (mainly meta-heuristics) proposed over the last decade [29, 4, 5, 3].

Not preparing the project execution based on a schedule that can more or less be trusted (but rather, letting the realized schedule unfold during execution) has been recognized as a shortcoming of S-RCPSP [16]. This shortcoming motivates us and a number of other authors to propose a proactive-reactive approach, with [31, 11, 21] yielding the most promising computational results in existing literature. Different approaches pursue different optimization objectives; however, the common aim is to optimize some *tradeoff* between expected makespan and expected deviation from the proactive schedule. In line with other authors we represent activity duration distributions with a sample and propose a Linear Programming (LP)-based heuristic for PS-RCPSP, a Mixed-Integer LP (MILP) model enabling us to obtain exact solutions, and a MILP-based heuristic which asimilates *iterative flattening* [27]. The (MI)LP models for PS-RCPSP proposed in this paper are the result of adjusting the models proposed by

Artigues et al. in [2] and [23]. We refer to exact solutions assuming stochastic duration distributions can be represented exactly by a sufficiently large sample.[1]

The contributions of this paper extend from section 3 and onwards. In order to base the paper on a consistent and self-contained framework of notation, section 2 summarizes existing concepts from deterministic, reactive, and proactive-reactive project scheduling. Section 3 introduces in a formal manner the problem studied here, that we name Proactive Stochastic RCPSP. Section 4 presents one main contribution of this paper: a LP-based heuristic for solving this problem. Section 5 presents another main contribution: a MILP model for this problem which enables us to obtain exact PS-RCPSP solutions. To our knowledge, no other exact solution methods have been proposed for poblems of similar type. Section 6 presents yet another contribution, a MILP-based heuristic for PS-RCPSP. Section 7 presents an experimental study in which we find that the LP-based heuristic performs favorably in comparison to the state-of-art, especially when the aim is to achieve near-zero instability. We also find the MILP-based heuristic to be more effective (albeit less efficient) when one is willing to accept medium levels of instability in order to minimize expected makespan. Section 8 concludes the paper.

2 Preliminaries

The purpose of this section is to introduce the research area of proactive-reactive (stochastic) project scheduling, which is where the contributions of this paper belong to. To establish autonomy and to facilitate discussion in further sections, we use convenient notation (which sometimes departs from standard notation) and begin with summarizing existing concepts from deterministic and (purely) reactive project scheduling. For a comprehensive survey of deterministic, reactive and proactive-reactive project scheduling, the reader may refer to [17].

2.1 Deterministic project scheduling

A project is usually represented as a directed acyclic graph $G(N, E)$, with nodes $N = \{1, \ldots, n\}$ corresponding to the set of n project activities. Each directed arc (i, j) in $E \subseteq \{(i, j) \in N^2\}$ defines a direct temporal constraint between activities i and j, meaning that j may not start unless activity i has finished. In effect, E defines a binary, irreflexive and transitive relation: if there is a path from activity i to activity j in $G(N, E)$ then j cannot start unless i has finished. Let us $T(E) \supseteq E$ denote the transitive closure of E, defined as

$$T(E) := \{(i, j) \in N^2 : \exists \text{ a path from } i \text{ to } j \text{ in } G(N, E)\}$$

We shall name a *temporally independent set* each subset of activities $X \subseteq N$ which are mutually independent with respect to temporal constraints. That is, if X is a temporally independent set, then $X^2 \cap T(E) = \emptyset$. Obviously, if only temporal constraints are taken into account, the activities of a temporally independent set may overlap in time in a schedule.

[1] This notion of exactness is in line with Stork [29] who represent stochastic duration distributions with a sample when proposing exact search methods for the S-RCPSP.

We assume as input a set $R := \{1, \ldots, m\}$ of m resources which must be shared among activities. Each resource $r \in R$ is associated with known capacity $b_r \in \mathbb{N}_0$. Furthermore, each activity $i \in N$ requires a known amount $q_{ir} \leq b_r$ of resource r while it executes. Vector $\boldsymbol{b} \in \mathbb{N}_0^m$ and matrix $\boldsymbol{q} \in \mathbb{N}_0^{n \times m}$ define the problem's resource constraints. Every independent set X for which $\sum_{i \in X} q_{ir} > b_r$ for some $r \in R$ is called a *forbidden* set. Even though it is allowed by the temporal constraints E, all activities in X may not overlap at some timepoint t because resource r will be used beyond its capacity, which is not possible.

Let $H \subseteq N^2$ denote a set of temporal constraints. Below we give the definition of a function Φ which returns the set of all forbidden sets w.r.t. temporal constraints H and the problem's resource constraints $(\boldsymbol{q}, \boldsymbol{b})$.

$$\Phi(H) := \{X \subseteq N : X^2 \cap T(H) = \emptyset, \sum_{i \in X} q_{ir} > b_r \text{ for some } r \in R\} \tag{1}$$

In addition to the parameters mentioned so far, we assume as input a vector $\boldsymbol{d} \in \mathbb{N}_0^n$ such that d_i defines the duration of activity i. Overall, a tuple $(N, R, E, \boldsymbol{d}, \boldsymbol{q}, \boldsymbol{b})$ specifies an instance of the RCPSP. A schedule $\boldsymbol{s} \in \mathbb{N}_0^n$ such that s_i defines the start time of activity i, is a feasible solution when it satisfies the temporal and resource constraints, meaning that

$$s_j \geq s_i + d_i \qquad \forall (i,j) \in E \tag{2}$$

$$a(\boldsymbol{s}, t) \notin \Phi(E) \qquad \forall t \geq 0 \tag{3}$$

Here, $a(\boldsymbol{s}, t) := \{i \in N : t \in [s_i, s_i + d_i)\}$ is the set of activities executing at timepoint t according to $\boldsymbol{s}$ and Φ as defined earlier. Thus, (3) ensures there is no timepoint t at which the activities of a forbidden set overlap concurrently.

Project source-sink convention. RCPSP asks to find a feasible schedule of minimum makespan $C_{\max}(\boldsymbol{s}) := \max\{s_i + d_i : i \in N\}$. Most RCPSP-related works assume that the last activity, n, is a dummy activity with zero duration (i.e. $d_n = 0$) and that it must wait for the completion of every other activity (i.e. $(i, n) \in T(E)$ for every $i \in N - \{n\}$). This dummy activity is often known as the project "sink" and it holds that $C_{\max}(\boldsymbol{s}) = s_n$. Another convention of most RCPSP-related works is that the first activity, 1, often known as the project "source" must be waited on by every other activity (i.e. $(1, j) \in T(E)$ for every $j \in N - \{1\}$).

We shall hereafter assume activities 1 and n correspond to the project source and sink, respectively. The RCPSP can now be formally stated as:

$$\boldsymbol{s}^* := \arg\min\{s_n : (2), (3), \boldsymbol{s} \geq 0\} \tag{4}$$

2.2 Reactive project scheduling

In the research area of stochastic project scheduling, the activity durations vector $\boldsymbol{d}$ is replaced with a stochastic vector $\boldsymbol{D}$ such that D_i is the stochastic variable representing the uncertain duration of activity i, with a known probability distribution $\mathbb{P}[D_i = t]$. In line with recent works on S-RCPSP, we shall denote (elements of) stochastic vectors with a capital symbol.

S-RCPSP is a purely reactive extension of RCPSP. The solution sought for is no longer a schedule, but a reactive scheduling policy. A policy is a combinatorial object π which parameterizes the mapping from stochatic vector $\boldsymbol{D}$ to a corresponding realized schedule $\boldsymbol{S}(\pi, \boldsymbol{D})$. Note that $\boldsymbol{S}$ denotes a function which returns a vector of activity start times (of length n). Furthermore, if a realization $\boldsymbol{d}$ of $\boldsymbol{D}$ is passed as an argument, then $\boldsymbol{S}(\pi, \boldsymbol{d})$ denotes a deterministic vector. if $\boldsymbol{D}$ is passed as an argument, $\boldsymbol{S}(\pi, \boldsymbol{D})$ denotes a stochatic vector.

Different classes of policies have been proposed in the literature [25,26,29,3]. One condition that all policy classes are expected to satisfy is that function $\boldsymbol{S}$ complies with the *non-anticipativity constraint*: the decision to start activity i at time $[\boldsymbol{S}(\pi, \boldsymbol{D})]_i$ cannot rely on information from the feature: the value of $[\boldsymbol{S}(\pi, \boldsymbol{D})]_i$ must be determined by time $t \leq [\boldsymbol{S}(\pi, \boldsymbol{D})]_i$. Other features such as the structure of π and the definition of function $\boldsymbol{S}$ depend on the class under study.

List-based policies. Two classes of policies which are prominent in the literature are *resource-based* (rb) policies and *activity-based* (ab) policies, also known collectively as *list-based policies.* A list-based policy is a priority vector $\boldsymbol{l} \in \mathbb{R}^n$ assigning priority l_i to activity i. Realized schedule $\boldsymbol{S}(\boldsymbol{l}, \boldsymbol{D})$ is computed by a variant of the well-known parallel schedule-generation-scheme (SGS) complying with the non-anticipativity constraint [5]; with the SGS definition being slightly different between rb-policies and ab-policies. As far as list-based policies are concerned, S-RCPSP asks to find a vector $\boldsymbol{l} \in \mathbb{R}^n$ that minimizes $\mathbb{E}[[\boldsymbol{S}(\boldsymbol{l}, \boldsymbol{D})]_n]$. Stork [29] proposes exact branch-and-bound algorithms for both rb and ab-policies. Ballestín [4] proposes an efficient genetic algorithm for ab-policies, providing the first computational experience on larger S-RCPSP instances. Ballestín and Leus [5] manage to obtain better results with a Greedy Randomized Adaptive Search Procedure (GRASP), again for the class of ab-policies. The best performance (w.r.t. expected makespan minimization) has so far been obtained with the more recent work of Ashtiani et al. [3] who propose a GRASP for a new class, namely *pre-processing* (pp) policies–a hybrid between rb-policies and es-policies.

Earliest-start policies. An es-policy is a set of temporal constraints $\mathcal{E} \subseteq N^2$ chosen such that

$$T(\mathcal{E}) \supseteq E, \tag{5}$$

$$\Phi(\mathcal{E}) = \emptyset, \tag{6}$$

$$G(N, \mathcal{E}) \text{ is acyclic} \tag{7}$$

Condition (5) ensures that a schedule $\boldsymbol{s}$ satisfying $s_j \geq s_i + d_i$ for each $(i, j) \in \mathcal{E}$ (here $\boldsymbol{d}$ can be any arbitrary choice of activity durations) is feasible with respect to the problem's precedence constraints E. Condition (6) ensures that $\boldsymbol{s}$ satisfying $\mathcal{E}$ implies that it also satisfies resource constraints prescribed by availabilities $\boldsymbol{b}$ and requirements $\boldsymbol{q}$ (as described earlier). Condition (7) ensures that the set of schedules satisfying $\mathcal{E}$ (for any arbitrary choice of activity durations $\boldsymbol{d}$) is non-empty.

When a project is executed according to an es-policy $\mathcal{E}$, the schedule that is realized, $\boldsymbol{S}(\mathcal{E}, \boldsymbol{D})$, is what is often known as the *earliest-start* schedule specified by $\mathcal{E}$. The earliest-start schedule of $\mathcal{E}$ can be defined as:

$$[\boldsymbol{S}(\mathcal{E}, \boldsymbol{D})]_j := \max\{[\boldsymbol{S}(\mathcal{E}, \boldsymbol{D})]_i + D_i : (i, j) \in \mathcal{E}\} \tag{8}$$

To put it simply, an activity j starts immediatelly when all its predecessors in $G(N, \mathcal{E})$ have finished. This time quantity (the latest finish time of j's predecessors) is often known as the length of the *critical path* from project source 1 to activity j. As far as es-policies are concerned, the S-RCPSP asks to find some $\mathcal{E}^*$ which minimizes $[\boldsymbol{S}(\mathcal{E}, \boldsymbol{D})]_n$ (the length of the critical path to the project sink) by expectation:

$$\mathcal{E}^* := \arg\min\{\mathbb{E}[[\boldsymbol{S}(\mathcal{E}, \boldsymbol{D})]_n] : (5-7), \mathcal{E} \in N^2\} \tag{9}$$

Constraints $(5-7)$ enforce the feasibility of any realized schedule with both the precedence and the resource constraints of the problem.

Stork [29] proposed an exact branch-and-bound search for problem (9). His algorithm considers each *minimal* forbidden set X (subset-minimal forbidden set) in some order and branches on each of $|\{(i,j) \in X^2\}|$ arcs which can be included in $\mathcal{E}$ in order to eliminate X from $\Phi(\mathcal{E})$. Without obtaining new computational results, in [22] Leus gives a formal treatment of es-policies as resource-flow networks (flow networks which can represent feasible RCPSP schedules) and proposes a refined version of the branch & bound algorithm of Stork. Exploiting the relation between resource-flows and es-policies, Artigues et al. [23] propose a robust optimization model for es-policies, for when a stochastic characterization of uncertainty is not available.

2.3 Proactive-reactive project scheduling

Reactive project scheduling allows one to pick activity start times dynamically during the project, under conditions of uncertainty. A main drawback of this approach (e.g. [16, 17, 9]) is that prior to (and during) project execution there is no schedule prescribing activity start times that can more or less be trusted. Such a "proactive" schedule can serve important organizational purposes; in fact, the deviation of the realized schedule from this proactive schedule is expected to induce organizational costs.

Attempts to overcome this drawback gave rise to the research area of proactive-reactive project scheduling, which is the research area that this paper belongs to. The main idea behind the proactive-reactive approach is to execute the project by using a proactive schedule together with a scheduling policy. Under uncertainty, some activities may not start at their proactive start times, because activities they have to wait for are not yet finished and/or resources they require are not yet released. In such cases, the scheduling policy determines which activities to start at their prescribed start times and which to postpone. It should be noted that most works assume *railway-mode scheduling*, meaning that an activity may not start earlier than its proactive start time, which strengthens the "stability" of the project execution. Clearly, the realized schedule is a function of the policy and the proactive schedule. Achieving low instability (deviation of the realized from the proactive schedule) requires "spreading" proactive activity start times far appart, in effect increasing the expected makespan. The general aim is to optimize some tradeoff between expected makespan and expected instability.

Van de Vonder et al. [32, 31, 30] propose and evaluate experimentally various proactive-reactive heuristics. The proposed heuristics assume as input an instance of S-RCPSP along with an initial schedule. The best performing heuristic is the so-called Starting Time Criticality (STC) heuristic. An es-policy is extracted from the structure of this initial schedule and used to iteratively transform the initial schedule into a proactive schedule by inserting time-buffers betwen activities. Deblaere et al. [11] propose an approach which integrates the proactive step (forming a proactive schedule) and the reactive step (forming the adjoining policy). Their approach is only possible to compare with ours and others that assume railway-mode scheduling, by choosing sufficiently high penalties for earliness (w.r.t. the proactive schedule).[2] More recently, Vilches and Demeulemeester [21] propose a Chance-Constrained Programming model (CCP) for the RCPSP which asks to find a minimum makespan schedule subject to probabilistic temporal and resource constraints. They propose a Mixed Integer LP model, the solution to which is a proactive schedule that will most likely remain feasible under stochastic duration variability, without presumption on the policy that will be used during project execution.

3 Proactive Stochastic RCPSP

In deterministic and reactive project scheduling, the main problem under study (RCPSP and S-RCPSP, respectively) is stated clearly. A clearly stated problem model cannot be found in proactive-reactive project scheduling literature, perhaps because this research area is still in a burn-in phase.[3]Existing literature seems to pursue the general aim of optimizing some tradeoff between expected makespan and expected instability (deviation from the proactive schedule). This section presents the formal statement of a proactive-reactive scheduling problem for which (heuristic and exact) solution methods are proposed in subsequent sections.

The problem presented here, the Proactive Stochastic RCPSP (PS-RCPSP), asks to find a tuple $(\mathcal{E}, \boldsymbol{t})$ where $\mathcal{E}$ is an es-policy and $\boldsymbol{t}$ is a proactive schedule, minimizing the weighted sum of two performance criteria:

1. expected value of project makespan,
2. expected value of tardiness with respect to proactive release-times.

The first criterion measures lack of robustness and is relevant for obvious reasons. The second criterion measures instability and captures the expected deviation of the realized schedule from the proactive schedule. Note that $\boldsymbol{t}$ prescribes activity release-times (an activity i may not start earlier than t_i). Intuitively, instability represents the extent to which the proactive start-times can be trusted, when used for organizational purposes before and during project execution.

An instance of this problem is encoded by a tuple $(n, m, \boldsymbol{q}, \boldsymbol{b}, E, \boldsymbol{D}, \alpha)$. For clarity, we summarize the meaning of problem parameters. Positive integer n is the number of activities and m is the number of resources. Parameters $\boldsymbol{q} \in \mathbb{N}_0^{m \times n}$ and $\boldsymbol{b} \in \mathbb{N}_0^m$ define resource requirements and availabilities respectively. Set $E \subseteq \{1, \ldots, n\}^2$ defines pairwise precedence constraints. Stochastic vector $\boldsymbol{D}$ is of length n with each element D_i a stochatic variable (with given distribution $\mathbb{P}[D_i = t]$) which describes the uncertain duration of activity i. Finally, parameter $\alpha \in [0, 1]$ determines the desired tradeoff between robustness (i.e. minimization of expected makespan) and stability (i.e. minimization of expected instability). More emphasis can be put on either minimizining makespan (by choosing α closer to one) or minimizing instability (by choosing α closer to zero).

[2] We are grateful to one of our anonymous reviewers for this remark.

[3] Some works refer to [16] but this is a formal treatment of a proactive-reactive *machine* scheduling problem.

Formally, the problem can be stated as follows:

$$
\begin{aligned}
\min \quad & f(\mathcal{E}, \boldsymbol{t}) := \alpha \cdot \mathbb{E}\left[[\boldsymbol{S}((\mathcal{E}, \boldsymbol{t}), \boldsymbol{D})]_n\right] + (1-\alpha) \cdot \mathbb{E}\left[\sum_{i=1}^{n}([\boldsymbol{S}((\mathcal{E}, \boldsymbol{t}), \boldsymbol{D})]_i - t_i)\right] && (10)\\
\text{s.t.} \quad & \Phi(G(N, \mathcal{E})) = \emptyset && (11)\\
& T(\mathcal{E}) \supseteq E && (12)\\
& G(N, \mathcal{E}) \text{ is acyclic} && (13)\\
& \mathcal{E} \in \{1, \ldots, n\}^2, \boldsymbol{t} \geq 0 && (14)
\end{aligned}
$$

Conditions (12,13) ensure there is a non-empty set of schedules satisfying the problem's precedence constraints as prescribed in E. Condition (11) ensures that each such schedule also satisfies the problem's resource constraints prescribed by $\boldsymbol{q}$ and $\boldsymbol{b}$.

4 Heuristic LP-based approach

This section presents a polynomial-time heuristic for PS-RCPSP which consists of two steps:

1. using mean activity durations, the deterministic RCPSP $(n, m, \boldsymbol{q}, \boldsymbol{b}, E, \mathbb{E}[\boldsymbol{D}])$ is solved to obtain a good schedule $\boldsymbol{s}$ and a feasible es-policy $\mathcal{E}$ (i.e. satisfying (11),(12) and (13)) is derived from the structure of $\boldsymbol{s}$ in polynomial time (this procedure is described in [2]),
2. by solving a linear program presented below, we find a proactive schedule $\boldsymbol{t}$ that is optimally combined with $\mathcal{E}$ (which is kept fixed) so as to minimize an approximation of (10).

After $\mathcal{E}$ has been obtained in the first step, finding $\boldsymbol{t}$ which minimizes (an approximation of) the PS-RCPSP objective is achieved by solving the LP model presented below.

$$
\begin{aligned}
\min \quad & \left[\alpha\left(\frac{1}{|\Gamma|}\sum_{\gamma\in\Gamma} s_n^{\gamma}\right) + (1-\alpha)\left(\frac{1}{|\Gamma|}\sum_{i=1}^{n}\sum_{\gamma\in\Gamma}(s_i^{\gamma} - t_i)\right)\right] && (15)\\
\text{s.t.} \quad & s_j^{\gamma} \geq s_i^{\gamma} + d_i^{\gamma} \qquad \forall (i,j) \in \mathcal{E}, \gamma \in \Gamma && (16)\\
& s_i^{\gamma} \geq t_i \qquad \forall i = 1, \ldots, n && (17)\\
& \boldsymbol{t} \geq 0 && (18)
\end{aligned}
$$

Here, (15) approximates the objective (10) based on $\Gamma \subseteq \mathbb{R}^n$: an adequately-sized sample of stochastic vector $\boldsymbol{D}$. The realization of activity durations under sample scenario $\gamma \in \Gamma$ is represented by vector $\boldsymbol{d}^{\gamma} = (d_1^{\gamma}, \ldots, d_n^{\gamma})$. The corresponding realized schedule is *earliest-start*$((\mathcal{E}, \boldsymbol{t}), \boldsymbol{d}^{\gamma}) = (s_1^{\gamma}, \ldots, s_n^{\gamma})$, as computed by the model constraints. The solution is a proactive schedule $\boldsymbol{t} = (t_1, \ldots, t_n)$ that optimizes the tradeoff between expected makespan and instability for the given es-policy $\mathcal{E}$. This LP model has $n(|\Gamma| + 1)$ linear variables (n variables t_i and $n|\Gamma|$ variables s_i^{γ}).

4.1 Related work

Van de Vonder et al. [31] propose several heuristics, of which the most competitive is the Starting Time Criticality (STC) heuristic and we shall therefore restrict our attention to it. Our LP-based heuristic bears similarities with STC. In fact, the first step of our heuristic is identical to that of STC: an es-policy $\mathcal{E}$ is extracted by the structure of an initial schedule $\boldsymbol{s}$. The second step of STC involves transforming the "unstable" schedule $\boldsymbol{s}$ into a "stable" schedule $\boldsymbol{t}$ with an iterative procedure, while keeping $\mathcal{E}$ fixed. In each iteration a one-unit time buffer is added at the start of that activity that "needs it the most" (as determined by a proposed "starting time criticality" measure) until adding more buffer time would not further reduce the instability of $\boldsymbol{t}$, which is measured by

$$\mathbb{E}\left[\sum_{i=1}^{n} w_i([\boldsymbol{S}((\mathcal{E},\boldsymbol{t}),\boldsymbol{D})]_i - t_i)\right] \tag{19}$$

Here, w_i is a cost associated with the instability of activity i. Furthermore, t_n is kept fixed to a project deadline and therefore w_n represents the marginal cost of deviating from this project deadline. Note that by replacing α in (10) with individual weights w_i and choosing a fixed project deadline, it is straightforward to adapt our approach to the instability objective considered by van de Vonder et al. However, we felt that the choice of (10) as an objective is advantageous, as it underlines the tradeoff between expected makespan and instability more clearly and simplifies discussion by not involving a weight per individual activity and not requiring the choice of a project deadline.

Note that $\boldsymbol{t}$ is not guaranteed to be (precedence and resource) feasible with respect to mean activity durations (as required in the work of van de Vonder [31]). Enforcing $\boldsymbol{t}$ to hold this property in our approach can be accomplished by including the following constraint in the LP model:

$$t_j \geq t_i + \mathbb{E}[D_i] \qquad \forall (i,j) \in \mathcal{E}$$

However, this property only adds to the organizational value of $\boldsymbol{t}$ when mean values are reasonable estimates of activity durations.

Let us note that both our heuristic and STC have polynomial worst-case complexity (in the number of activities). However, in contrast with STC, our approach guarantees that $\boldsymbol{t}$ is chosen optimally when $\mathcal{E}$ is kept fixed and assuming the distribution of $\boldsymbol{D}$ is approximated with a sample. Therefore, if efficiency considerations enable us to choose a large-enough sample Γ (which is mostly the case due to the efficiency of existing LP solvers), our heuristic is expected to perform at least as well as STC. Finally, note that our heuristic is simpler to implement, requiring only the description of the presented LP model.

Leus et al. [24] assume as input a proactive schedule $\boldsymbol{t}$ (e.g. one that has been produced by STC). They propose a branch-and-bound search which returns the es-policy $\mathcal{E}$ which fits $\boldsymbol{t}$ optimally in minimizing an expression of expected instability similar to (19).

5 Exact MILP-based approach

PS-RCPSP (section 3) asks to find an es-policy and a proactive schedule $(\mathcal{E}, \boldsymbol{t})$ that together minimize the weighted sum of expected makespan and instability. Section 4 presented a heuristic approach according to which $\mathcal{E}$ is kept fixed while $\boldsymbol{t}$ is optimally paired with the policy by solving a LP. This section presents a Mixed Integer LP (MILP) model with which PS-RCPSP can be solved to optimality. However, it should be pointed-out that a solution is trully exact only if we assume stochastic duration distributions can be accurately described by the chosen sample Γ; for general probability distributions we obtain a lower bound. In fact, the problem of computing the exact expected makespan of a given es-policy (and assuming duration distributions with discrete support) has been shown by Hagstrom in [13] to be intractable. However, our notion of exactness is in line with the computational study of Stork [29] where "optimal" scheduling policies are computed by using a fixed sample of duration distributions.

This model includes binary variables representing the structure of $\mathcal{E}$ and linear variables representing $\boldsymbol{t}$. To our knowledge, no other exact approaches have been proposed in the literature for problems of similar type (i.e. asking for a scheduling policy and proactive schedule that together optimize some tradeoff between expected makespan and instability). To arrive at this PS-RCPSP model, we merge the LP model presented in the previous section with a MILP model that has been proposed by Artigues et al. [2] and which allows to solve the deterministic RCPSP by treating it as a flow-network problem. The model presented here is not entirely new, since a similar technique (repeating precedence constraints for each scenario of the chosen sample) has been proposed in [23] for minimizing the maximum regret of an es-policy.

5.1 The RCPSP model of Artigues et al.

Artigues et al. [2] represent a solution to the RCPSP as a so-called *resource-flow* $\boldsymbol{f} \in \mathbb{R}_0^{n\times n\times m}$; an assignment to variables f_{ijr} associated with each pair of activities $(i,j) \in N^2$ and each resource $r \in R$. A resource-flow describes the "passing" of resource units inbetween activities. More precisely, $\boldsymbol{f}$ is an indirect representation of every schedule $\boldsymbol{s}$ in which f_{ijr} units of resource r are released by activity i at its completion $s_i + d_i$ and then "picked up" by activity j at its start s_j, without another activity using these units between $s_i + d_i$ and s_j.

A resource-flow is feasible when it satisfies

$$\sum_{j\in N-\{i\}} f_{jir} = q_{ir} \quad \forall i \in N - \{1\} \tag{20}$$

$$\sum_{j\in N-\{i\}} f_{ijr} = q_{ir} \quad \forall i \in N - \{n\} \tag{21}$$

Eq. (20) asks that each activity i (except for the sink) receives as many resource units as it requires the moment it starts. Eq. (21) asks that each activity i (except for the source) releases as many resource units as it has used the moment it finishes.

The flow network $G(N, \phi(\boldsymbol{f}))$ associated with $\boldsymbol{f}$ is defined as $\phi(\boldsymbol{f}) := \{(i,j) \in N^2 : f_{ijr} > 0 \text{ for some } r \in R\}$; i.e. there is an arc from each activity to every other activity it passes at least one resource unit to. As shown by Leus [22,23], feasible resource-flows and es-policies are interrelated: $\mathcal{E} = E \cup \phi(\boldsymbol{f})$ is a feasible es-policy if $\boldsymbol{f}$ is a feasible resource-flow (and $G(N,\mathcal{E})$ is acyclic). Therefore, every schedule which satisfies $G(N,\mathcal{E})$ is feasible. The following MILP model proposed by Artigues et al. enables one to find a feasible resource-flow $\boldsymbol{f}$ which minimizes the cost (described by function g) of a schedule $\boldsymbol{s}$ which satisfies the temporal constraints of $G(N, E \cup \phi(\boldsymbol{f}))$.

$$\begin{aligned}
\min \quad & s_n && (22)\\
\text{s.t.} \quad & s_j \geq s_i + d_i - M(1 - z_{ij}) \qquad \forall (i,j) \in N && (23)\\
& z_{ij} = 1 \qquad \forall (i,j) \in E && (24)\\
& f_{ijr} \leq M z_{ij} \qquad \forall (i,j) \in N^2, r \in R && (25)\\
& (20), (21) && (26)\\
& f_{ijr} \geq 0, z_{ij} \in \{0,1\} \qquad \forall (i,j) \in N^2, r \in R && (27)
\end{aligned}$$

Here M is a large constant. Due to (26) $\boldsymbol{f}$ is a feasible resource-flow. Due to (25), if $f_{ijr} > 0$ for one or more $r \in R$ then $z_{ij} = 1$, meaning that variables z_{ij} describe the flow-network $\phi(\boldsymbol{f})$ of the resource-flow (i.e. $\phi(\boldsymbol{f}) = \{(i,j) \in N^2 : z_{ij} = 1\}$). Due to (23) and (24), $\boldsymbol{s}$ describes a schedule which satisfies the temporal constraints in $G(N, E \cup \phi(\boldsymbol{f}))$. Since $\boldsymbol{f}$ is a feasible resource-flow, $\boldsymbol{s}$ is a feasible schedule.

5.2 Extension for S-RCPSP

This section presents a trivial extension to the RCPSP model of Artigues et al. which enables us to find optimal es-policies for the S-RCPSP. Considering a sample $\Gamma \subset \mathbb{R}^n$ of stochastic activity durations vector $\boldsymbol{D}$ allows us to present the following MILP model.

$$\begin{aligned}
\min \quad & \frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} s_n^\gamma && (28)\\
\text{s.t.} \quad & s_j^\gamma \geq s_i^\gamma + d_i^\gamma - M(1 - z_{ij}) \qquad \forall (i,j) \in N^2, \gamma \in \Gamma && (29)\\
& (24 - 27) && (30)
\end{aligned}$$

Our extension is rather straightfoward. Each variable s_i is included here as variable s_i^γ for each sample scenario $\gamma \in \Gamma$. Precedence constraints (23) from before are now replicated for each scenario $\gamma \in \Gamma$ in condition (29). Objective (22) is now replaced with objective (28), which estimates the makespan expectation $\mathbb{E}[S(E \cup \phi(\boldsymbol{f}))]$ based on sample Γ.

5.3 Extension for PS-RCPSP

Here we extend the previous model by including a variable t_i for each $i \in N$, which determines the activity's proactive starting time. The resulting PS-RCPSP MILP model is presented below.

$$\begin{aligned}
\min \quad & \left[\alpha \left(\frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} s_n^\gamma \right) + (1 - \alpha) \left(\frac{1}{|\Gamma|} \sum_{i=1}^{n} \sum_{\gamma \in \Gamma} (s_i^\gamma - t_i) \right) \right] && (31)\\
\text{s.t.} \quad & s_j^\gamma \geq s_i^\gamma + d_i^\gamma - M(1 - z_{ij}) \qquad \forall (i,j) \in N^2, \gamma \in \Gamma && (32)\\
& (24 - 27) && (33)\\
& s_i^\gamma \geq t_i \qquad i \in N, \gamma \in \Gamma && (34)\\
& \boldsymbol{t} \geq 0 && (35)
\end{aligned}$$

The objective now becomes identical to that of the LP-based heuristic, measuring the weighted sum of expected makespan and expected instability. Condition (34) ensures that an activity may not start earlier than its proactive start time.

To summarize, by solving this model we obtain a PS-RCPSP solution $(\mathcal{E}, \boldsymbol{t})$ where $\mathcal{E} = \{(i,j) \in N^2 : z_{ij} = 1\}$ is a feasible es-policy and $\boldsymbol{t}$ defines a proactive schedule.

Proposition 1 *Define $\mathcal{E} := \{(i,j) : z_{ij} = 1\}$ the arcs of the flow-network $G(N, E \cup \phi(\boldsymbol{f}))$ associated with resource-flow $\boldsymbol{f}$. Let $\boldsymbol{f}, \boldsymbol{z}, \boldsymbol{s}, \boldsymbol{t}$ be an optimal solution. For each scenario $\gamma \in \varGamma$, $\boldsymbol{s}^\gamma$ defines a schedule where each activity i starts as soon as allowed by its proactive release-time t_i and es-policy $\mathcal{E}$.*

Proof For each scenario $\gamma \in \varGamma$, let $\bar{x}^\gamma$ denote the earliest allowed start time for activity i, allowed by the combination of $\mathcal{E}$ and proactive schedule $\boldsymbol{t}$. We want to prove that in an optimal solution, $\boldsymbol{s}^\gamma = \bar{x}^\gamma$ for all $\gamma \in \varGamma$.

Assume that $s_i^\gamma = \bar{x}_i^\gamma + \delta$ for some $\gamma \in \varGamma, i \in N$, with $\delta > 0$. Since (31) increases monotonically with s_i^γ, the objective can be improved by setting $s_i^\gamma = \bar{x}_i^\gamma$, without violating any constraints. Therefore, in every optimal solution we have $s_i^\gamma = \bar{x}_i^\gamma$ for all $\gamma \in \varGamma$ and $i \in N$, meaning that each $\boldsymbol{s}^\gamma$ defines the earliest start times schedule allowed by the combination of by es-policy $\mathcal{E}$ and proactive schedule $\boldsymbol{t}$ under scenario γ. $\square$

By proposition 1 it follows that an optimal solution to the MILP model presented above is, in fact, an optimal solution for the PS-RCPSP.

6 Heuristic MILP-based approach

Even for small instances (e.g. with 30 activities and 4 resources), solving the proposed model might take an inordinate amount of time. We propose an algorithm (Algorithm 1), the main idea of which was inspired by the *iterative flattening* heuristic of Oddi et al. [27]. The heuristic of Oddi et al. was developed for the deterministic RCPSP with minimum/maximum time-lag precedence constraints [15]. Every feasible schedule for the problem they study is compactly represented as a network of temporal constraints (known as a Simple Temporal Network [12]). It is the similarity with an es-policy (which is a network of zero-lag temporal constraints) that has inspired the development of the heuristic presented here.

The proposed heuristic involves solving a sequence of sufficiently small subproblems with non-increasing optimal objective values. Each iteration involves solving a partially solved instance to optimality. Thus, worst-case complexity is exponential in the number of activities. In practice, however, "good" solutions can be obtained with relative efficiency.

Algorithm 1 assumes as input an instance of the PS-RCPSP. According to aforementioned notation, the instance is represented as $(N, R, E, \boldsymbol{D}, \boldsymbol{q}, \boldsymbol{b}, \alpha)$. An initial solution is obtained by solving a deterministic RCPSP (lines 1-3) which can be done efficiently with one of the various existing heuristics. This solution will serve as a starting point for the first iteration, which is described as follows. A partial solution is formed by removing a random subset of highly critical arcs from the current solution (line 6). The resulting subproblem is solved to optimality (by use of the proposed model) and a complete solution is obtained (line 7). If this new solution is better, it becomes the starting point of the next iteration. The algorithm may terminate when, e.g., a chosen number of iterations have been performed, or the objective has failed to improve a certain number of times.

Algorithm 1 Iterative flattening for PS-RCPSP

```
 1: $\boldsymbol{s} \leftarrow$ schedule for RCPSP $(N, R, E, \mathbb{E}[\boldsymbol{D}], \boldsymbol{q}, \boldsymbol{b})$
 2: $\mathcal{E}^* \leftarrow E \cup \phi(\boldsymbol{f^s})$ with $\boldsymbol{f^s}$ extracted from $\boldsymbol{s}$
 3: $\boldsymbol{t}^* \leftarrow (0, \ldots, 0)$
 4: while termination criteria not met do
 5:   $\mathcal{H} \leftarrow$ random subset of $\mathcal{E}^* - T(E)$ chosen by criticality probability
 6:   $(\mathcal{E}, \boldsymbol{t}) \leftarrow$ optimal solution for PS-RCPSP $(N, R, \mathcal{E}^* - \mathcal{H}, \boldsymbol{D}, \boldsymbol{q}, \boldsymbol{b}, \alpha)$
 7:   if $(\mathcal{E}, \boldsymbol{t})$ has a lower objective than $(\mathcal{E}^*, \boldsymbol{t}^*)$ then
 8:     $(\mathcal{E}^*, \boldsymbol{t}^*) \leftarrow (\mathcal{E}, \boldsymbol{t})$
 9:   end if
10: end while
11: return $(\mathcal{E}^*, \boldsymbol{t}^*)$
```

Further efficiency improvements. Note that the optimal solution $(\mathcal{E}, \boldsymbol{t})$ of the subproblem solved in each iteration cannot be worse than the best solution seen so far, $(\mathcal{E}^*, \boldsymbol{t}^*)$. To improve performance one may use $(\mathcal{E}^*, \boldsymbol{t}^*)$ as an initial solution when solving the model (line 6). Efficiency can be further improved by reducing the number of binary variables z_{ij} in the model. This can be accomplished by observing that z_{ij} for each $(i, j) \in T(\mathcal{E}^* - \mathcal{H})$ can be fixed to one and z_{ij} for each $(j, i) \in T(\mathcal{E}^* - \mathcal{H})$ can be fixed to zero.

7 Experiments

In [21], Vilches and Demeulemeester compare their method (CCP) with that by Van de Vonder et al. (STC) [31]. To the best of our knowledge, STC and CCP constitute the state-of-art as far as trading expected makespan for instability in stochastic project scheduling is concerned. In this section we extend this comparison by using the same experimental set-up and including results for our LP-based heuristic (Section 4) and the MILP-based heuristic (Section 6). As the results show, our approaches compare favorably with STC and CCP.

The set-up used in [21] was based on the `J30` deterministic RCPSP bench-set of the well-known PSPLIB [19], which comprises 480 deterministic RCPSP instances, each with $n = 30$ activities. Based on this bench-set, three stochastic RCPSP bench-sets were derived, namely `J30-low`, `J30-med`, and `J30-high`, corresponding to conditions of low, medium, and high project uncertainty, with activity durations following a discretized beta distribution. Specifically, each activity i with duration d_i in the deterministic RCPCP instance now has stochastic duration $D_i = [X_i d_i 0.5(l + h)]$ with $\mathbb{E}[D_i] \simeq d_i$ where

- X_i follows a beta distribution with shape parameters $\alpha = 2, \beta = 5$;
- $l = 0.75$ and $h = 1.625$ in the low variability bench-set;
- $l = 0.5$ and $h = 2.25$ in the medium variability bench-set;
- $l = 0.25$ and $h = 2.875$ in the high variability bench-set;
- operator $[\cdot]$ represents rounding to the closest integer.

Each of the evaluated methods (including STC and CCP) has a certain "tradeoff parameter" which determines whether more emphasis is put on minimizing expected makespan or minimizing expected instability. For our LP-based and MILP-based heuristic this tradeoff parameter is the weight α in expression (10). CCP and STC have corresponding parameters with a similar effect. By varying the choice of the

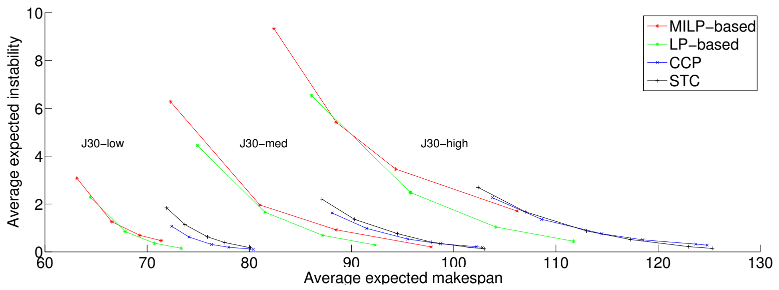

Fig. 1: Trading expected makespan for stability.

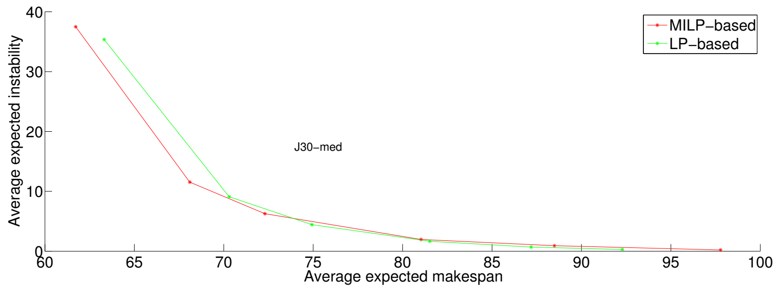

Fig. 2: Trading expected makespan for stability for higher α.

corresponding tradeoff parameter(s), a set of tradeoff data-points is obtained for each method, on each of the three bench-sets.

In Figure 1, the data-points for each method are displayed as a tradeoff curve, on each of the three bench-sets, resulting in three "clusters" of tradeoff curves. A tradeoff curve captures the average performance of the method on that bench-set. Specifically, each data-point is two-dimensional and records the average expected makespan and average expected instability for a certain choice of the tradeoff parameter(s), where the average is taken over all 480 instances of the bench-set. Data-points for CCP and STC are borrowed from the work of Vilches and Demeulemeester [21]. Data-points for our heuristics are obtained by setting $\alpha = 0.05$, 0.1, 0.2, and 0.4. Higher alpha values correspond to data-points closer to the upper left corner, with higher instability and lower makespan.

Figure 2 focuses on the medium variability case for higher α values, including additional data-points for $\alpha = 0.6$ and $\alpha = 0.9$. This allows us to compare the MILP-based and LP-based heuristics when more emphasis is put on minimizing expected makespan.

The expected makespan and expected instability of the solution provided by each of the methods on a particular instance is computed with a sample $\Gamma^{large} \subset \mathbb{R}^n$ comprising $|\Gamma^{large}| = 10^3$ realization of durations vector $\boldsymbol{D}$. Note that the data-points of Vilches and Demeulemeester were computed with a different sample of size 10^3. We assume that 10^3 is a sufficiently large sample size to facilitate comparability with our results.

Configuration of heuristics. The sample Γ^{milp} used by our MILP-based heuristic during optimization (see line 6 of Algorithm 1) is of size $|\Gamma^{milp}| = 30$. Our MILP-based heuristic is configured to perform three (3) iterations and the number of highly critical arcs removed in each iteration (see line 5 of Algorithm 1) is $|\mathcal{H}| = 20$. Note that the criticality of the arcs is computed based on sample Γ^{large} (this is done efficiently, in time quadratic in n and linear in $|\Gamma^{large}|$). The solver we use is CPLEX version 12.6. Furthermore, we set a time-limit for the solver to 50 seconds (since each iteration starts from a feasible solution, the solver will always return with a solution within the time-limit). The polynomial-time complexity of our LP-based heuristic (no binary variables in the model) allows us to use a large sample during optimization. In fact, we use sample Γ^{large}. To find a deterministic schedule (as required in step 1 of this LP-based heuristic) we used a priority rule procedure recently proposed in [10]. Vilches and Demeulemeester use a sample of size 10^2 during optimization, for both STC and CCP. Furthermore, they limit the time spent in solving their CCP model on an instance to a maximum of 10 seconds.

Observations. Figure 1 suggests that regardless of the mode of variability (low, medium, or high), when the purpose is to achieve near-zero instability, the LP-based heuristic yields the best results. This can be attributed to the efficiency of solving a LP model, which enables us to use a large sample (of size 10^3 in this case) during optimization.

Figure 2 suggests that even though the sample used during optimization is much smaller for the MILP-based heuristic (of size 30), it is more effective than the LP-based heuristic for higher α values (i.e. when minimizing instability is more important than minimizing makespan). Both the LP-based and the MILP-based heuristics start from the same es-policy (see step 1 in section 4 and line 2 of Algorithm 1, respectively). However, the MILP-based heuristic restructures the policy and this enables it to perform better at minimizing expected makespan.

Restructuring the policy comes at the cost of efficiency. With three iterations allowed per instance, this yields an average of 50 seconds per instance for the MILP-based heuristic. The LP-based heuristic is considerably more efficient, with an average of 1.5 seconds per instance. Vilches and Demeulemeester report that STC spends on average 0.2 seconds per instance, while their CCP approach spends on average 10 seconds per instance.

8 Conclusions and future work

This paper proposes the PS-RCPSP problem model which, assuming stochastic activity durations, asks to find a so-called earliest-start (es) policy and a proactive schedule that together minimize the weighted sum of expected project makespan and expected instability. Extending an existing MILP model for the RCPSP, a MILP model for PS-RCPSP is presented, which allows us to find optimal (es-policy, proactive schedule) pairs. Solving this problem to optimality might require an impractical amount of time, even for instances with few activities (e.g. 30). Therefore, we propose a LP-based and a MILP-based heuristic for the PS-RCPSP. Our LP-based heuristic optimizes the proactive schedule by keeping the es-policy part of the solution fixed. Our MILP-based heuristic optimizes the structure of the policy together with the proactive schedule. The LP-based heuristic, which is rather efficient, seems to be more effective compared to the state-of-art (i.e. achieves smaller expected makespan for a certain level of expected instability) especially when the aim is to achieve close to zero instability. The MILP-based heuristic is rather effective when the aim is to achieve low expected makespan at the cost of moderate or high instability. In contrast to existing state-of-art approaches such as CCP [21] and STC [31], our heuristics rely on the idea of optimizing the proactive schedule together with the scheduling policy. This difference might in part explain observed performance differences.

Future work involves a thorough experimental analysis of the proposed heuristics, not for the purpose of comparing them to the state-of-art, but for a deeper understanding of their behavior and its dependence on problem characteristics. Furthermore, most existing stochastic project scheduling works are evaluated on instances where the deterministic RCPSP counterpart instance (formed by mean activity durations) serves as a good approximation of the stochastic instance. This is exploited by our heuristics and other heuristics such as STC. However, in certain practical domains (e.g. maintenance scheduling), the duration of some activities is known a-priori with accuracy, while the duration of other activities follows a distribution with very high variance. In maintenance scheduling, for example, "inspection" activities have known durations but "repair" activities might be (un)necessary with certain probabilities. We would like to investigate performance on such instances which cannot be approximated well by their determinitic counterpart.

Acknowledgements We would like to thank our anonymous reviewers for their constructive remarks. This research belongs to the *Job Scheduling Problem* part of the *Rolling Stock Life Cycle Logistics* applied research and development programme, funded by *NS/NedTrain*.

References

1. Christian Artigues, Sophie Demassey, and Emmanuel Neron. *Resource-constrained project scheduling: models, algorithms, extensions and applications*. John Wiley & Sons, 2013.
2. Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249–267, 2003.
3. Behzad Ashtiani, Roel Leus, and Mir-Bahador Aryanezhad. New competitive results for the stochastic resource-constrained project scheduling problem: exploring the benefits of pre-processing. *Journal of Scheduling*, 14(2):157–171, 2011.
4. Francisco Ballestín. When it is worthwhile to work with the stochastic rcpsp? *Journal of Scheduling*, 10(3):153–166, 2007.
5. Francisco Ballestin and Roel Leus. Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management*, 18(4):459–474, 2009.
6. J-H Bartels and Jürgen Zimmermann. Scheduling tests in automotive r&d projects. *European Journal of Operational Research*, 193(3):805–819, 2009.
7. Jacek Blazewicz, Jan Karel Lenstra, and AHG Rinnooy Kan. Scheduling subject to resource constraints: classification and complexity. *Discrete Applied Mathematics*, 5(1):11–24, 1983.
8. Felix Bomsdorf and Ulrich Derigs. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. *Or Spectrum*, 30(4):751–772, 2008.
9. Kristof Braeckmans, Erik Demeulemeester, Willy Herroelen, and Roel Leus. Proactive resource allocation heuristics for robust project scheduling. *DTEW Research Report 0567*, pages 1–22, 2005.
10. Frits de Nijs and Tomas Klos. A novel priority rule heuristic: Learning from justification. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.
11. Filip Deblaere, Erik Demeulemeester, and Willy Herroelen. Proactive policies for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, 214(2):308–316, 2011.

12. Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.
13. Jane N Hagstrom. Computational complexity of pert problems. *Networks*, 18(2):139–147, 1988.
14. Sönke Hartmann and Dirk Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207(1):1–14, 2010.
15. Willy Herroelen, Erik Demeulemeester, and Bert De Reyck. 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, 2001.
16. Willy Herroelen and Roel Leus. The construction of stable project baseline schedules. *European Journal of Operational Research*, 156(3):550–565, 2004.
17. Willy Herroelen and Roel Leus. Robust and reactive project scheduling: a review and classification of procedures. *International Journal of Production Research*, 42(8):1599–1620, 2004.
18. Rainer Kolisch and Sönke Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European journal of operational research*, 174(1):23–37, 2006.
19. Rainer Kolisch and Arno Sprecher. Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European Journal of Operational Research*, 96(1):205–216, 1997.
20. Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based milp models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):3–13, 2011.
21. Patricio Lamas and Erik Demeulemeester. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling*, pages 1–20, 2015.
22. Roel Leus. Resource allocation by means of project networks: dominance results. *Networks*, 58(1):50–58, 2011.
23. Roel Leus, Christian Artigues, and Fabrice Talla Nobibon. Robust optimization for resource-constrained project scheduling with uncertain activity durations. In *Industrial Engineering and Engineering Management (IEEM), 2011 IEEE International Conference on*, pages 101–105. IEEE, 2011.
24. Roel Leus and Willy Herroelen. Stability and resource allocation in project planning. *IIE transactions*, 36(7):667–682, 2004.
25. Rolf H Möhring, Franz Josef Radermacher, and Gideon Weiss. Stochastic scheduling problems i – general strategies. *Zeitschrift für Operations Research*, 28(7):193–260, 1984.
26. Rolf H Möhring, Franz Josef Radermacher, and Gideon Weiss. Stochastic scheduling problems ii – set strategies. *Zeitschrift für Operations Research*, 29(3):65–104, 1985.
27. Angelo Oddi and Riccardo Rasconi. Iterative flattening search on rcpsp/max problems: Recent developments. In *Recent Advances in Constraints*, pages 99–115. Springer, 2009.
28. Andreas Schutt, Thibaut Feydy, Peter J Stuckey, and Mark G Wallace. Solving rcpsp/max by lazy clause generation. *Journal of Scheduling*, 16(3):273–289, 2013.
29. Frederik Stork. Branch-and-bound algorithms for stochastic resource-constrained project scheduling. *Technical rep*, pages 702–2000, 2000.
30. Stijn Van de Vonder, Francisco Ballestin, Erik Demeulemeester, and Willy Herroelen. Heuristic procedures for reactive project scheduling. *Computers & Industrial Engineering*, 52(1):11–28, 2007.
31. Stijn Van de Vonder, Erik Demeulemeester, and Willy Herroelen. Proactive heuristic procedures for robust project scheduling: An experimental analysis. *European Journal of Operational Research*, 189(3):723–733, 2008.
32. Stijn Van de Vonder, Erik Demeulemeester, Willy Herroelen*, and Roel Leus. The trade-off between stability and makespan in resource-constrained project scheduling. *International Journal of Production Research*, 44(2):215–236, 2006.