MISTA 2015

Trading robustness for stability in stochastic project scheduling – a mathematical programming approach

First Author · Second Author

Abstract If you are submitting a full paper, insert your abstract here. If you are NOT submitting a full paper (i.e. an abstract or extended abstract), then delete this section

1 Introduction

Project scheduling literature largely concentrates on finding a schedule subject to temporal and resource constraints. The schedule sought for is an assignment of start-times to activities and serves an important function: it facilitates the efficient and coordinated use of limited resources, in order to minimize a lateness measure such as the project makespan. Finding such a schedule usually invovles solving an instance of the resource-constrained project scheduling problem (RCPSP) [9], which has been shown to be NP-Hard [cite]. RCPSP finds several industrial applications (e.g. [6,5]) and associated literature is quite extensive, with numerous exact and (meta-)heuristic algorithms. The literature lacks heuristics with performance guarantees but existing (meta-)heuristics are able to find good solutions for large instances and various lateness measures.

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. Task durations used for preparing the schedule are mostly rough estimates, since most projects are subject to delays during execution. 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. If these modifications are decided ad-hoc, realized activity start-times might differ significantly from planned start-times, compromising project timeliness in unpredictable ways.

This paper addresses the issue of hedging against uncertainty, by preparing a schedule in combination with a project execution strategy, assuming activity durations

First Author
Affiliation
E-mail: Second.Author@example.com

Second Author
Affiliation
E-mail: Second.Author@example.com

are stochastic variables with known probability distributions. We propose a mathematical programming model along with an associated heuristic for a generalization of RCPSP that we name the Proactive Stochastic RCPSP (PS-RCPSP).

We introduce this problem as a generalization of the Stochastic RCPSP (S-RCPSP), a problem already known in the area of stochastic project scheduling [10]. S-RCPSP asks to find a *stochastic scheduling policy* instead of a schedule. Various *classes* of policies have been proposed in the literature [16, 17, 10] but in general, a policy is a combinatorial object representing a set of rules that may guide the materialization of a final realized schedule, as actual activity durations become known during project execution. In effect, the policy defines a mapping from every possible realization of activity durations to a corresponding realized schedule. S-RCPSP asks to find a policy that minimizes the expected value of some project lateness measure (e.g. the expected project makespan). Few exact and heuristic S-RCPSP solution algorithms have been developed over the last decade [20, 3, 4, 2].

One drawback of S-RCPSP is that no schedule (that can more or less be trusted) is prescribed prior to project execution and activity start-times are determined in a purely reactive manner. The problem studied here, the PS-RCPSP, asks to find a *proactive* schedule paired with an *earliest-start* (es) policy. An es-policy is a set of temporal constraints between pairs of activities. The proactive schedule prescribes release-times (i.e. earliest possible start-times) for activities. A activity may start at its prescribed release-time or later in case of delays (i.e. other activities requiring more than allocated time) as determined by the es-policy. In effect, the tuple (proactive schedule, es-policy) defines a mapping from every possible realization of activity durations to a corresponding realized schedule. PS-RCPSP asks to find this tuple that minimizes the *weighted sum* of two performance criteria:

1. expected value of project lateness (e.g. expected project makespan),
2. expected value of tardiness with respect to proactive release-times.

The first criterion, to which we shall refer as *lack of robustness*, is relevant for obvious reasons. The second criterion, to which we shall refer as *lack of stability* or *nervousness*, captures the expected deviation of the realized schedule from the proactive schedule and represents the extent to which the proactive start-times can be trusted. One typically wants to achieve low lack of stability in order to avoid what has been known as *shop-floor nervousness* [19].

The content of this paper and contributions therein are structured as follows. Section 3 presents the definition of PS-RCPSP as a generalization of S-RCPSP. In Section 5 we present a straightforward extension of the flow-network model for RCPSP proposed by Artigues et al. [1] into the first (to our knowledge) mathematical programming model for S-RCPSP. This allows us to present an extended model for PS-RCPSP, which (in contrast to S-RCPSP) is a continuous optimization problem. The state-of-art for problems similar to PS-RCPSP [23, 22, 8, 12] optimize the proactive schedule and/or the scheduling policy separately. Although not as efficient, solving our model enables the optimization of the proactive schedule together with the policy, making it the first exact method for proactive stochastic project scheduling. Section 6 proposes a heuristic which combines the idea of *iterative flattening* [18] with the PS-RCPSP model, for solving small to medium-sized instances efficiently. This heuristic allows to arbitrarily calibrate the tradeoff between optimality and efficiency. Section 7 presents an experiment with small instances of PSPLIB [11]. We find that our heuristic mostly outperforms the state-of-art in proactive stochastic project scheduling.

2 Preliminaries (needs compaction)

2.1 RCPSP

A project is usually represented as a direced 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.

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 resource constrained project scheduling problem (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. RCPSP asks to find a feasible schedule of minimum makespan $C_{max}(\boldsymbol{s}) := \max\{s_i + d_i : i \in N\}$ which can be formally stated as

$$\boldsymbol{s}^* := \arg\min\{C_{max}(\boldsymbol{s}) : (2), (3)\} \tag{4}$$

2.2 Stochastic RCPSP

In the research area of stochastic project scheduling, the activity durations vector $\boldsymbol{d}$ is replaced with a stochastic vector $\boldsymbol{\mathcal{D}}$ such that D_i is the stochastic variable representing the uncertain duration of activity i, with a known probability distribution $\mathbb{P}[D_i = t]$.[1]

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 p from every possible duration realization $\boldsymbol{d}$ of stochatic vector $\boldsymbol{\mathcal{D}}$ to a corresponding realized schedule $p(\pi, \boldsymbol{d}) \in \mathbb{R}_0^n$. Mapping p must comply with the *non-anticipativity constraint*: the decision to start activitty i at time $[p(\pi, \boldsymbol{d})]_i$ cannot rely on information from the feature; it must have been taken by time $t \leq [p(\pi, \boldsymbol{d})]_i$.

Different classes of policies have been proposed in the literature [16, 17, 20, 2]. The form of π and the definition of p depend on the class of the policy.

2.2.1 Earliest-start policies

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

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

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

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

For every realization $\boldsymbol{d}$ of $\boldsymbol{\mathcal{D}}$ and due to (5), a schedule $\boldsymbol{s}$ that satisfies $\mathcal{E}$ (i.e. $s_j \geq s_i + d_i$ for each $(i, j) \in \mathcal{E}$) also satisfies temporal constraints E. For every realization $\boldsymbol{d}$ of $\boldsymbol{\mathcal{D}}$ and due to (6), a schedule $\boldsymbol{s}$ satisfying $\mathcal{E}$ also satisfies resource constraints $(\boldsymbol{q}, \boldsymbol{b})$. Eq. (7) ensures that the set of schedules which satisfy $\mathcal{E}$ (for every possible choice of $\boldsymbol{d}$) is non-empty.

Projected schedule $p(\mathcal{E}, \boldsymbol{d})$ is computed as the *earliest-start* schedule of $\mathcal{E}$ under scenario $\boldsymbol{d} \in \Omega$, meaning that

$$[p(\mathcal{E}, \boldsymbol{d})]_j := \max\{[p(\mathcal{E}, \boldsymbol{d})]_i + d_i : (i, j) \in \mathcal{E}\} \tag{8}$$

That is, to execute the earliest-start schedule means to start each activity j immediatelly when all its predecessors in $G(N, \mathcal{E})$ have finished. This quantity is known in the literature as the duration of the *critical path* from source activity 1 to activity j in $G(N, \mathcal{E})$. Since durations are stochastic the duration of the critical path to j, $[p(\mathcal{E}, \boldsymbol{\mathcal{D}})]_j$ is also stochastic. That is, p returns a stochastic vector schedule $\mathcal{X} = p(\mathcal{E}, \boldsymbol{\mathcal{D}})$ and the cost of this schedule $g(\mathcal{X})$ (i.e. its makespan $\max\{[p(\mathcal{E}, \boldsymbol{\mathcal{D}})]_i : i \in N\}$) is also a stochastic quantity. S-RCPSP asks to find an optimal es-policy which can be defined as

$$\mathcal{E}^* := \arg\min\{\mathbb{E}[g(p(\mathcal{E}, \boldsymbol{\mathcal{D}}))] : (5, 6, 7), \mathcal{E} \in N^2\} \tag{9}$$

Stork [20] has proposed an branch-and-bound algorithm for problem (9). His algorithm considers each *minimal* forbidden set X (subset-minimal forbidden set) in some order and branches on each of $|X^2|$ arcs which can be included in $\mathcal{E}$ in order to eliminate X from $\Phi(\mathcal{E})$. Without obtaining new computational results, in [13] Leus

[1] We shall use the S-RCPSP notation proposed in [2], denoting (elements of) stochastic vectors with a capital letter.

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, Leus and Artigues [14] propose a robust optimization model for es-policies, for when a stochastic characterization of uncertainty is not available.

2.2.2 List-based policies

For completeness, we also discuss the classes of *resource-based* (rb) policies and *activity-based* (ab) policies. An rb-policy is a priority vector $\boldsymbol{l} \in \mathbb{R}^n$ assigning priority l_i to activity i. Projected schedule $p(\boldsymbol{l}, \boldsymbol{d})$ is computed by a variant of the well-known parallel schedule-generation-scheme (SGS) complying with the non-anticipativity constraint [4]. At $t = 0$ and at each subsequent activity completion $t > 0$ one starts as many activities as allowed by temporal and resource constraints. A similar class is that of *activity-based* (ab) policies; an ab-policy is again a priority vector. But the behavior of p is now slightly different: a activity may not start at t (even if temporal and resource constraints allow this) unless all activities of lower priority have finished by t. Note that every projected schedule $p(\boldsymbol{l}, \boldsymbol{d}), \boldsymbol{d} \in \Omega$ is feasible regardless of the choice of $\boldsymbol{l}$ (due to procedure p). S-RCPSP then asks to find a vector $\boldsymbol{l} \in \mathbb{R}^n$ that minimizes $\mathbb{E}[g(p(\boldsymbol{l}, \boldsymbol{\mathcal{D}}))]$.

Stork [20] proposes exact branch-and-bound algorithms for both rb and ab-policies. Ballestín [3] proposes an efficient genetic algorithm for ab-policies, providing the first computatinal experience on larger S-RCPSP instances. Ballestín and Leus [4] 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. [2] who propose a GRASP for a new class, namely *pre-processing* (pp) policies–a hybrid between rb-policies and es-policies.

3 Proactive Stochastic RCPSP (work in progress)

Having a schedule prior to project execution (that can more or less be trusted) serves important functions (e.g. the organization of resources and negotiation of contracts [10, 7]). Therefore, not producing a schedule has been recognized as a major drawback of the S-RCPSP. So-called "proactive-reactive" scheduling methods [10, 15, 22, 8, 12] attempt to overcome this drawback by proposing that the output of the scheduling effort is a pair $(\pi, \boldsymbol{t})$. Here, π defines a scheduling policy and $\boldsymbol{t}$ a proactive schedule. Each activity i starts either at its proactive start-time t_i or later (in case of delays) as dictated by policy π.

Let $p((\pi, \boldsymbol{t}), \boldsymbol{\mathcal{D}})$ denote the stochastic schedule that is realized accordingly. Deviations of this realized schedule from the proactive schedule $\boldsymbol{t}$ induce organizational costs. The expected deviation (sometimes known as lack of stability, or nervousness [19]) associated with $(\pi, \boldsymbol{t})$ is measured as

$$\sum_{i \in N} \mathbb{E}\{[p((\pi, \boldsymbol{t}), \boldsymbol{\mathcal{D}})]_i - t_i\} \tag{10}$$

Van de Vonder et al. [23, 22, 21] examine various two-stage approaches, the most competitive of which is the Start-Time Criticality (STC) heuristic. First, a deterministic

RCPSP schedule $\boldsymbol{t}^0$ is found (using mean activity durations). Then, an es-policy is extracted ad-hoc from the structure of $\boldsymbol{t}^0$ and is used as a guide to transform $\boldsymbol{t}^0$ into a stable schedule $\boldsymbol{t}$ by insertion of buffer-time between activities. Recently, Lamas and Demeulemeester [12] propose a chance-constrained programming (CCP) model for RCPSP, the solution to which is a stable proactive schedule. Leus and Herroelen [15] assume they are given such a stable proactive schedule $\boldsymbol{t}$ and focus on the reactive step: with a branch & bound search, an es-policy $\mathcal{E}$ is fitted to $\boldsymbol{t}$ such that $(\mathcal{E}, \boldsymbol{t})$ minimizes (10). Deblaere et al. [8] transform an input schedule $\boldsymbol{t}^0$ into a proactive schedule $\boldsymbol{t}$, at the same time choosing π from the class of rb-policies so as to minimize nervousness, in what seems to be the only approach which integrates the proactive and the reactive steps. However, they assume an activity i may start earlier than its prescribed start-time t_i (i.e. the stable schedule does not define release-times), rendering their approach not directly comparable.

A characteristic of existing approaches is that the proactive schedule $\boldsymbol{t}$ and policy π are not optimized together as a tuple $(\pi, \boldsymbol{t})$. Instead, either the policy (for [22,12]) or the proactive schedule (for [15]) are supposed to be chosen separately, without suggestions on how to do this effectively. In fact, this characteristic was recognized as a drawback in [22] by Van de Vonder et al. (since the structure of both π and $\boldsymbol{t}$ determine overall performance) which lead to a more integrated approach in [8].

Along similar lines with [8], we propose an integrated approach to proactive-reactive scheduling.

PS-RCPSP asks to find this tuple that minimizes the *weighted sum* of two performance criteria:

1. expected value of project lateness (e.g. expected project makespan),
2. expected value of tardiness with respect to proactive release-times.

In this paper we propose a mixed-integer linear programming (MILP) model which can be used to solve the following problem

$$(\mathcal{E}^*, \boldsymbol{t}^*) := \arg\min\{\alpha \cdot \mathbb{E}[g(p(\mathcal{E}, \boldsymbol{\mathcal{D}}))] + (1-\alpha) \cdot \Delta : (5,6,7), \mathcal{E} \in N^2, \boldsymbol{t} \in \mathbb{R}^n\} \quad (11)$$

where $[p((\mathcal{E}, \boldsymbol{t}))]_i$ denotes the start time of activity i, subject to temporal constraints $\mathcal{E}$ and such that it never starts earlier than t_i and $\Delta := \sum_{i \in N} \mathbb{E}\{[p((\mathcal{E}, \boldsymbol{t}), \boldsymbol{\mathcal{D}})]_i - t_i\}$.

4 Heuristic LP approach

This section presents a heuristic for PS-RCPSP which consists of two stages:

1. the deterministic RCPSP (using mean activity durations) is solved to obtain a good schedule $\boldsymbol{s}$ and an es-policy $\mathcal{E}$ is derived from $\boldsymbol{s}$ in polynomial time (as in [1]),
2. a proactive schedule $\boldsymbol{t}$ is optimally "fitted" to $\mathcal{E}$ so as to minimize

$$f(\boldsymbol{t}) := \alpha \mathbb{E}[[p((\mathcal{E}, \boldsymbol{t}), \boldsymbol{\mathcal{D}})]_n] + (1-\alpha) \sum_{i \in N} \mathbb{E}[[p((\mathcal{E}, \boldsymbol{t}), \boldsymbol{\mathcal{D}})]_i - t_i]$$

Here, parameter $\alpha \in [0,1]$ determines the desired tradeoff between expected makespan and instability. Finding the optimal $\boldsymbol{t}$ for the chosen $\mathcal{E}$ and α is achieved by solving the LP model presented below.

$$\min \qquad \hat{f}(\boldsymbol{t}) := \left[\alpha \left(\frac{1}{|\Gamma|} \sum_{\gamma \in \Gamma} s_n^\gamma \right) + (1-\alpha) \left(\frac{1}{|\Gamma|} \sum_{i=1}^{n} \sum_{\gamma \in \Gamma} (s_i^\gamma - t_i) \right) \right] \tag{12}$$

$$\text{s.t.} \qquad s_j^\gamma \geq s_i^\gamma + d_i^\gamma \qquad \forall (i,j) \in \mathcal{E}, \gamma \in \Gamma \tag{13}$$

$$s_i^\gamma \geq t_i \qquad \forall i = 1, \ldots, n \tag{14}$$

Here, $\Gamma \subseteq \mathbb{R}^n$ is a sample (of adequate size) of stochastic vector $\boldsymbol{\mathcal{D}}$, used in estimating expected values in the objective. 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 $p((\mathcal{E}, \boldsymbol{t}), \boldsymbol{d}^\gamma) = (s_1^\gamma, \ldots, s_n^\gamma)$, 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 $\mathcal{E}$. More emphasis can be put on either minimizing makespan or minimizing instability by choosing α. This LP model has n linear variables t_i and $n \cdot |\Gamma|$ linear variables s_i^γ.

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 [22]). Including the following constraint enforces $\boldsymbol{t}$ to hold this property.

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

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

5 Exact mixed integer LP approach

Artigues et al. [1] treat RCPSP as a flow-network problem. The solution is represented as a resource flow. They propose a MILP model. Leus study the equivalence between a resource flow and an es-policy. In their work with Artigues they extend this model with robust optimization techniques. We will show that the model of Artigues can be extended in order to find optimal es-policies for the S-RCPSP (just a sketch, needs re-writing).

5.1 The RCPSP model of Artigues et al.

Artigues et al. [1] 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

$$\sum_{j \in N - \{i\}} f_{jir} = q_{ir} \quad \forall i \in N - \{1\} \tag{15}$$

$$\sum_{j \in N - \{i\}} f_{ijr} = q_{ir} \quad \forall i \in N - \{n\} \tag{16}$$

Eq. (15) asks that each activity i (except for the sink) receives as many resource units as it requires the moment it starts. Eq. (16) 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 [13,14], 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 $G(N, E \cup \phi(\boldsymbol{f}))$.

$$
\begin{array}{lll}
\text{min.} & g(\boldsymbol{s}) & \\
\text{s.t.} & f_{ijr} \leq M z_{ij} & \forall (i,j) \in N^2, r \in R \quad (i) \\
& s_j \geq s_i + d_i - M(1 - z_{ij}) & \forall (i,j) \in N^2 \quad (ii) \\
& z_{ij} = 1 & \forall (i,j) \in E \quad (iii) \\
& f_{ijr} \geq 0, z_{ij} \in \{0,1\} & \forall (i,j) \in N^2, r \in R \quad (iv) \\
& (15),\ (16) & (v)
\end{array}
$$

Here M is a large constant. Due to constraint (v) $\boldsymbol{f}$ is a feasible resource-flow. Due to (i), if $f_{ijr} > 0$ for some $r \in R$ then $z_{ij} = 1$. Thus, 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 (ii) and (iii), $\boldsymbol{s}$ 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

Define the indexing set $S := \{1, \ldots, |\Omega|\}$ where Ω is a sample of stochastic activity durations vector $\boldsymbol{\mathcal{D}}$. This section proposes a trivial extension to the aforementioned model which enables us to find optimal es-policies for the S-RCPSP (i.e. accounting for stochastic activity durations).

$$
\begin{array}{lll}
\text{min.} & \mathbb{E}[C] & \\
\text{s.t.} & s^s_j \geq s^s_i + d^s_i - M(1 - z_{ij}) & \forall (i,j) \in N^2, s \in S \quad (ii') \\
& (15),\ (16),\ (i),\ (iii),\ (iv) &
\end{array}
$$

For each scenario indexed by $s \in S$ the corresponding schedule is an assignment to variables $(s^s_1, \ldots, s^s_n)$. Moreover, the objective now becomes to minimize the expected cost

$$\mathbb{E}[C] = \frac{1}{|\Omega|} \sum_{s \in S} g(\boldsymbol{s}^s) \tag{17}$$

where g is a (linear) cost function such as the makespan.

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 objective now accounts for nervousness along with expected cost.

$$
\begin{array}{lll}
\text{min.} & \alpha \cdot \mathbb{E}[C] + (1-\alpha) \cdot \mathbb{E}[\Delta] & \\
\text{s.t.} & s_i^s \geq t_i & \forall i \in N, s \in S \quad (v) \\
& \boldsymbol{t} \in \mathbb{R}^n & (vi) \\
& (15),\ (16),\ (i),\ (ii'),\ (iii),\ (iv) &
\end{array}
$$

Here, α is a parameter controlling the tradeoff between expected cost and nervousness. Δ measures nervousness as the total expected tardiness, defined as

$$\mathbb{E}[\Delta] := \frac{1}{|\Omega|} \sum_{i \in N} \sum_{s \in S} (s_i^s - t_i) \tag{18}$$

Proposition 1 *Define $E_z := \{(i,j) : z_{ij} = 1\}$ 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 index $s \in S$, $\boldsymbol{s}^s$ is the earliest-start schedule of $G(N, E_z)$ but such that a activity i never starts before t_i (i.e. $\boldsymbol{t}$ prescribes proactive release-times).*

Proof Let $\bar{\boldsymbol{s}}^s$ denote the earliest-start schedule of $G(N, E_z)$ for $s \in S$ but such that $\bar{s}_i^s \geq t_i$. Assume that $s_i^s = \bar{x}_i^s + \delta$ for some $s \in S, i \in N$, with $\delta > 0$. But $\mathbb{E}[\Delta]$ increases monotonically with $\boldsymbol{s}^s$ which contradicts $\boldsymbol{s}^s$ being an optimal assignment in combination with $\boldsymbol{f}, \boldsymbol{z}, \boldsymbol{t}$. □

By Proposition 1 it follows that an optimal solution defines a pair $(\mathcal{E}, \boldsymbol{t})$ (where $\mathcal{E} := \{(i,j) : z_{ij} = 1\}$) which optimizes the tradeoff between expected cost and nervousness. To our knowledge this is the first exact proactive-reactive scheduling method which integrates the proactive and the reactive steps.

Moreover, this model constitutes a unified framework for scheduling with and without uncertainty, encompassing the deterministic RCPSP, the S-RCPSP and the PS-RCPSP. E.g., as an alternative to Stork's branch-and-bound method [20], one may choose $\alpha = 1$ to find optimal es-policies for the S-RCPSP.

6 Iterative flattening heuristic

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) inspired by the iterative flattening heuristic of Cesta et al. []. The proposed algorithm 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 $(N, R, E, \boldsymbol{\mathcal{D}}, \boldsymbol{q}, \boldsymbol{b}, \alpha)$. 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 $(N, R, E, \mathbb{E}[\boldsymbol{\mathcal{D}}], \boldsymbol{q}, \boldsymbol{b})$
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: $\quad \mathcal{H} \leftarrow$ random subset of $\mathcal{E}^* - T(E)$ chosen by criticality probability
6: $\quad (\mathcal{E}, \boldsymbol{t}) \leftarrow$ optimal solution for PS-RCPSP $(N, R, \mathcal{E}^* - \mathcal{H}, \boldsymbol{\mathcal{D}}, \boldsymbol{q}, \boldsymbol{b}, \alpha)$
7: $\quad$ **if** $(\mathcal{E}, \boldsymbol{t})$ has a lower objective than $(\mathcal{E}^*, \boldsymbol{t}^*)$ **then**
8: $\quad\quad (\mathcal{E}^*, \boldsymbol{t}^*) \leftarrow (\mathcal{E}, \boldsymbol{t})$
9: $\quad$ **end if**
10: **end while**
11: return $(\mathcal{E}^*, \boldsymbol{t}^*)$

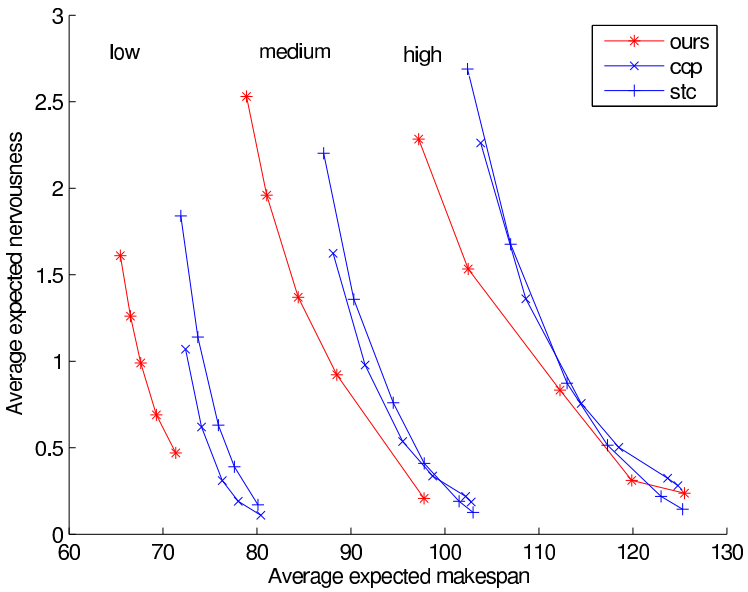

Fig. 1: Trading makespan for stability.

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). Furthermore, to reduce the number of binary variables z_{ij} in the model, one may let $z_{ij} = 1$ for each $(i, j) \in T(\mathcal{E}^* - \mathcal{H})$ and $z_{ij} = 0$ for each $(j, i) \in T(\mathcal{E}^* - \mathcal{H})$.

7 Experiments with PSPLIB

Acknowledgements Put any acknowledgements here, or delete this section

			makespan			nervousness		
variance	α	sample size	ours	ccp	stc	ours	ccp	stc
low	–	–	–	80.8	80.9	–	0.09	0.12
	0.05	50	71.1	80.4	80.1	0.49	0.11	0.17
	0.10	50	69.3	78.0	77.6	0.69	0.19	0.39
	0.15	50	67.4	76.3	75.9	0.92	0.31	0.63
	0.20	50	66.5	74.1	73.7	1.26	0.62	1.14
	0.25	50	65.0	72.4	71.9	1.59	1.07	1.84
medium	–	–	–	102.8	103.0	–	0.19	0.13
	0.05	50	97.8	102.2	101.5	0.21	0.22	0.19
	0.10	50	88.5	98.7	97.8	0.92	0.34	0.41
	0.15	50	84.4	95.5	94.5	1.37	0.54	0.76
	0.20	50	81.0	91.5	90.3	1.96	0.98	1.36
	0.25	50	65.0	78.9	87.1	2.53	1.62	2.20
high	–	–	–	124.8	125.3	–	0.28	0.15
	0.05	100	125.5	123.7	123.0	0.24	0.32	0.22
	0.10	100	119.9	118.5	117.3	0.31	0.50	0.52
	0.15	50	112.2	114.5	113.0	0.83	0.76	0.87
	0.20	50	102.5	108.6	107.0	1.53	1.36	1.68
	0.25	50	97.2	103.8	102.4	2.28	2.26	2.69

Table 1: Trading makespan for stability.

References

1. 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.
2. 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.
3. Francisco Ballestín. When it is worthwhile to work with the stochastic rcpsp? *Journal of Scheduling*, 10(3):153–166, 2007.
4. 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.
5. J-H Bartels and Jürgen Zimmermann. Scheduling tests in automotive r&d projects. *European Journal of Operational Research*, 193(3):805–819, 2009.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. Patricio Lamas Vilches and Erik Demeulemeester. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Available at SSRN 2464056*, 2014.

13. Roel Leus. Resource allocation by means of project networks: dominance results. *Networks*, 58(1):50–58, 2011.
14. 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.
15. Roel Leus and Willy Herroelen. Stability and resource allocation in project planning. *IIE transactions*, 36(7):667–682, 2004.
16. Rolf H Möhring, Franz Josef Radermacher, and Gideon Weiss. Stochastic scheduling problems i – general strategies. *Zeitschrift für Operations Research*, 28(7):193–260, 1984.
17. Rolf H Möhring, Franz Josef Radermacher, and Gideon Weiss. Stochastic scheduling problems ii – set strategies. *Zeitschrift für Operations Research*, 29(3):65–104, 1985.
18. Angelo Oddi and Riccardo Rasconi. Iterative flattening search on rcpsp/max problems: Recent developments. In *Recent Advances in Constraints*, pages 99–115. Springer, 2009.
19. Daniel C Steele. The nervous mrp system: how to do battle. *Production and Inventory Management*, 16(4):83–89, 1975.
20. Frederik Stork. Branch-and-bound algorithms for stochastic resource-constrained project scheduling. *Technical rep*, pages 702–2000, 2000.
21. 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.
22. 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.
23. 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.