Planning horizons based proactive rescheduling for stochastic resource-constrained project scheduling problems

Mario Brčić[a*], Marija Katić[b], Nikica Hlupić[c]

[a,c]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

[b]Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, United Kingdom

[a]mario.brcic@fer.hr,[b]m.katic@bbk.ac.uk,[c]nikica.hlupic@fer.hr

July 24, 2018

*Corresponding author

Abstract

Parties that collaborate on projects need to synchronize their efforts. For this reason they seek a decreased rescheduling variability of the time arrangements. Proactive–reactive scheduling is important in such situations. It predominantly achieves synchronization through a shared baseline schedule and deviation penalties. As the latter currently introduce an unrealistically high level of inflexibility, the solution methods never proactively update the baseline schedule. We propose threshold-based cost functions for the deviation penalties to enable a more realistic modeling of aspects of project collaboration. These functions introduce a greater degree of flexibility through the notion of *planning horizons* for the activities. This results in the possibility of profitable proactive changes to the baseline schedule. We present two metaheuristic approaches for the case of stochastic durations: rollout-based and iterative policy search. Both these approaches use such opportunities to achieve substantial cost–performance improvements in comparison to the best existing method. This enhancement comes at the price of an increased computational burden and the greater complexity of the solution space.

Keywords: Project scheduling, Proactive scheduling, Rollout policies, Approximate dynamic programming, Stochastic RCPSP

1 Introduction

Most of the research in the field of resource-constrained project scheduling problems (RCPSP) focuses on deterministic problems. The research into stochastic variants has intensified in the last 15 years, most often dealing with the durations of the activities as the only source of uncertainty. This research has been pushed in two directions: pure reactive and proactive–reactive scheduling. The surveys can be found in (Herroelen & Leus, 2005) and (Li & Womer, 2015).

Pure reactive scheduling is most often concerned with optimizing the expected schedule makespan or other *regular performance measures* under the assumption of known probability distributions of the random variables. The solution is not a static schedule as in the case of deterministic problems because the solution has to successfully deal with all the possible scenarios. Instead, the project is examined as a multistage decision process and the schedule is created in stages. The policies control the project execution to achieve the desired end (Möhring, Radermacher, & Weiss, 1984).

In addition to the necessity of dealing with uncertainties, real-world projects are rarely executed completely in-house due to the increase in the complexity of products and services. In such an environment, synchronization among project collaborators becomes important. However, pure reactive scheduling is very sensitive to the uncertainty realizations because the system reacts to them just-in-time. This means that no precise forecasts of future decisions can be made in order to enable meaningful synchronization between the project collaborators.

In *proactive–reactive scheduling*, the baseline schedule is created in the proactive phase, before the execution of the project. It must absorb as much of the execution variability as is seen to be appropriate according to the chosen robustness measure. Such a schedule is used as the central point for time-arrangements and information sharing between the collabo-

rators. During the execution of the project, the central authority can use the reactive procedure to improvise a solution based on the uncertainty realizations received from the project execution sites. However, cost penalties are incurred for deviations from the baseline schedule that guides the pre-execution planning and preparation of activities. The two commonly used notions of robustness in proactive scheduling are quality and solution robustness (Demeulemeester & Herroelen, 2011). Quality robustness aims to maximize the probability of completing the project in time. Solution robustness focuses on reducing the variability of the rescheduling, which then improves the fulfillment of the time-arrangements. These two criteria are usually in competition with each other, making the problems that combine them bi-criteria problems. This is usually resolved by scalarization, using monetary cost parameters (Brčić, Kalpić, & Katić, 2014). In the rest of this paper we assume the validity of such monetary scalarization.

Current research in proactive–reactive scheduling has succeeded in creating procedures for finding an initial, static baseline schedule of reasonable quality. Reactive procedures have been instrumental in carrying out activity starts during project execution. However, due to computational tractability, the majority of current approaches in reactive procedures are based on *open-loop policies*. Such policies are optimized at time zero using only the information available at the time, which makes them static. Information is made available during the project execution and it can be used to proactively change future time-arrangements and adapt activity starting in order to get better overall performance. *Closed-loop policies* are approaches based on dynamic programming that enable taking advantage of the incoming information (Li & Womer, 2015). Proactive baseline-rescheduling with predictive start times was done only in (Davari & Demeulemeester, 2017) under the assumption of fixed penalties for changes to the baseline schedule and using conflict-based policies, which leaves room for improvement.

In this paper, we focus on uncertain activity durations with known joint probability distribution. We present our work on a special class of robustness measures, *cost-based flexibility* (CBF)—a family where the rescheduling costs depend on the temporal distance of the changes in the baseline schedule (Brčić et al., 2014). The main contributions of this paper are:

- A new robustness measure, *threshold cost-based flexibility* (TCBF), which is a generalization of the existing schedule stability measure from Leus and Herroelen (2004). The motivation for this approach is that baseline predictions during the project execution can become not only suboptimal, but even infeasible, as demonstrated in Section 3.2. Existing schedule stability measures penalize all adjustments to the baseline regardless of their distance into the future, which makes adjustments ineffective. Our new setting enables modeling situations where near-future predictions need to be stable and far-future predictions are allowed more flexibility in adjustments. This enables the creation of new proactive-reactive methods that function as hybrids between the proactive-reactive and pure reactive approaches.

- Two metaheuristic approaches, *flexible rollout* and *iterative online simulation-based descent*, which are the first approaches to search within a policy class that does the non-conflict-based proactive rescheduling. Both methods are formalized under the general rollout framework from Goodson, Thomas, and Ohlmann (2017) by defining novel, complex decision rules. The rollout framework is utilized as a metaheuristic on top of the existing top-performing open-loop policy method from Deblaere, Demeulemeester, and Herroelen (2011). This is done in order to get a closed-loop policy that enables proactive rescheduling, of which the base heuristic itself is incapable. Rollout, as an approximate dynamic programming (ADP) approach, enables us to efficiently deal with the curses of dimensionality, especially regarding the substantially

increased dimensionality of the decision space.

- Theoretical and experimental demonstrations that the two metaheuristic approaches are more effective than the top-performing method from Deblaere et al. (2011).

The remainder of this paper is organized as follows: Section 2 gives an overview of related work. Section 3 provides a formal problem definition, giving a detailed description of the new robustness measure that brings possibilities for proactive rescheduling. Our solution methods are defined in Section 4, and Section 5 validates our ideas through an experimental comparison with the top-performing method. We offer some conclusions and sketch future research directions in Section 6.

2 Related Work

Under stochastic pure reactive approaches, Möhring et al. (1984) modelled a general stochastic scheduling problem as a stochastic dynamic program (SDP). Creemers (2015) presented a SDP-based model for finding globally optimal closed-loop policy for SRCPSP. Such exact procedures are intractable due to the curses of dimensionality, so research focuses on ADP methods (Bertsekas & Tsitsiklis, 1996; Powell, 2011). Stork (2001) used policy search on different open-loop policy families. Rostami, Creemers, and Leus (2016) introduced generalized pre-processor policy family that dominates over several commonly used policy families. Rollout algorithms, as one of the ADP approaches, were introduced in (Bertsekas, Tsitsiklis, & Wu, 1997; Bertsekas & Castanon, 1999). General rollout framework for SDPs was presented in (Goodson et al., 2017). Rollout was used for solving SRCPSP to minimize expected makespan in Li and Womer (2015).

According to Demeulemeester and Herroelen (2011, p. 159), the majority of proactive scheduling procedures described in the literature commonly

measure solution robustness using the stability measure defined in (Leus & Herroelen, 2004). It is expressed as the expected sum of the weighted absolute deviations of the realized schedule from the initial baseline schedule. We shall refer to this measure as the *symmetric stability measure*. There is a method (Lambrechts, 2007) for proactive rescheduling under uncertainty in resource availability that uses the symmetric stability measure, but it achieves only moderate results (Demeulemeester & Herroelen, 2011). The cause might lie in an inherent inflexibility in the objective function (Brčić et al., 2014).

An approach based on a generalization of the symmetric stability measure was proposed in Deblaere et al. (2011). We shall refer to that measure as the *asymmetric stability measure* because it introduces differing costs for negative and positive deviations. They presented an integrated procedure for proactive and reactive scheduling, which forms the basis for our work. A new family of open-loop execution policies, *resource-based policies with release times* (RPRT), was introduced. It is parametrized by the vector of priorities π and the vector of release times τ for non-dummy activities. The RPRT heuristic at each time-point t uses a parallel schedule generation scheme to start activities in order of the priorities π if their release times are greater than or equal to t. They also used an efficient method, based on the news-vendor problem, for finding the proactive baseline schedule in a local search procedure *simulation-based descent* (SBD). A substantial domination in terms of performance was shown in comparison to the STC+D procedure (Van de Vonder, Demeulemeester, Leus, & Herroelen, 2006) under fairly general conditions. To the best of our knowledge, this is the best performing solution. However, once established, the baseline schedule is static until the end of project execution, with all further decisions being exclusively the reactive starts of activity executions.

Davari and Demeulemeester (2017) proposed a method that enables

proactive rescheduling of the baseline schedule under uncertainty in activity durations. They defined proactive and reactive RCPSP (PR-RCPSP) where objective function extends on the symmetric stability measure and adds a penalty for each change to the baseline. The assumption is that the number of reactions is directly related to the robustness of the schedule and that it also damages business credibility of the contractor. This approach focuses on conflict-based policies that reschedule the baseline only if the conflict with the baseline schedule occurs. The solution is found by solving the SDP with an approach that is hit by the curses of dimensionality, as explained in Section 4.

Above listed research assumes availability of information about uncertainty in the form of probability distribution. There are methods that only assume that uncertainty takes values from known closed intervals. Artigues, Leus, and Nobibon (2013) used heuristic robust optimization to find pure reactive policy that minimizes the maximum absolute regret in projects with uncertain activity durations. Stability approach with regular measures in machine scheduling problems is used in (Y. N. Sotskov & Lai, 2012; Y. Sotskov & Werner, 2014). Digraphs are used instead of schedules with start times. This approach in offline procedure constructs a set of potentially optimal digraphs. Online procedure uses incoming information and the aforementioned set to construct the final schedule. In contrast, our stochastic approach uses baseline schedules with predictive start times and does not assume regularity of perfomance measure.

3 Problem Description

The problem under consideration here is the cost-based flexible stochastic RCPSP (CBF-SRCPSP) from Brčić et al. (2014), defined by a tuple $(V, E, \Omega, \mathcal{F}, p, R, B, D, \delta, J_{-1})$. $V = \{0, ..., n+1\}$ is a set of $n+2$ non-preemptive activities, 0 and $n+1$ being dummy start and end activities.

$V' = V \setminus \{0, n+1\}$ is the set of non-dummy activities. Precedence relations are defined as the transitive closure of relation $E \subset V \times V$, where an activity cannot be started before its predecessors are completed. 0 precedes and $n+1$ succeeds all other activities in V. $(\Omega, \mathcal{F}, p)$ is the probability space that encodes the information about the uncertainty. $\Omega \subset \mathbb{N}_0^{n+2}$ is a **bounded** sample space and $\mathcal{F} = 2^{\Omega}$ is the set of all possible events. p is the joint probability distribution of the activity durations and is represented by a random vector $\boldsymbol{d} : \Omega \to \mathbb{N}_0^{n+2}$, where the dummy activities have a duration of zero. $R = \{R_1, ..., R_r\}$ defines a set of r renewable resources and $\boldsymbol{B} \in \mathbb{N}^r$ is a vector of resource availabilities. Single-mode activity demands on resources are given by the matrix $\boldsymbol{D} \in \mathbb{N}_0^{(n+2)\times r}$, where $(\forall r \in R)(\forall i \in V) D_{i,r} \leq B_r$. $\delta \in \mathbb{N}_0$ is the project due date. J_{-1} is the objective function and it is defined in the following subsection.

3.1 MDP formulation

We model CBF-SRCPSP as an MDP model suitable for ADP. Stages $k = -1, ..., L$ are points where decisions can be made. Stage $k = -1$ is for offline calculations, before the project start. Other stages take place during the project execution. At each timepoint, stages take place in a sequence until the timepoint-terminating decision *"next"* is made. Then, the following stage takes place with the next timepoint. The number of stages L, though potentially infinite in the general case, is by definition bounded for terminating policies. In this paper, we work with terminating policies.

3.1.1 States

A state at stage k, $S_k = (t_k, A_k, U_k, s_k, \mathcal{H}_k)$ stores: the current time t_k, the set of active activities A_k, the set of unstarted activities U_k, the current baseline schedule s_k, and the base heuristic $\mathcal{H}_k$. The baseline schedule holds the realized start times of the started activities and the predictive start times

of the unstarted activities. The initial state is $S_{-1} = (-1, \varnothing, V, -, -)$ at the time $t = -1$ (prior to the project start) with baseline schedule and base heuristic unset ('–').

3.1.2 Decisions

Let $Feas(S_k) \subseteq U_k$ be a set of all precedence- and resource-feasible activities at the state S_k. Also, let $\mathcal{X}(S_k)$ be the set of feasible decisions at the state S_k. It is defined as:

$$\mathcal{X}(S_k) = \{(i_k, \xi_k, \bar{\mathcal{H}}_k) | i_k \in \{Feas(S_k) \cup' next'\}, (\forall j \in V \setminus U_k)\xi_j = s_j\}$$

i_k is the activity to be started, or the timepoint-terminating decision *"next"*. $\xi_k \in \mathbb{N}_0^{|V|}$ is the new baseline that must not reschedule already started activities, and $\bar{\mathcal{H}}_k$ is the new base heuristic. For the initial state S_{-1}, additional constraint is $i_{-1} =' next'$. The decision u_k made at state S_k is an element of $\mathcal{X}(S_k)$. Markov policy $\mu = (\delta^\mu_{-1}, ..., \delta^\mu_L)$ is a sequence of decision rules, where each decision rule $\delta^\mu_k(S_k) : S_k \to \mathcal{X}(S_k)$ specifies action choice from state S_k.

3.1.3 Transition process

Let $F_{k+1} \subseteq A_k$ be the random information that is revealed after making the decision u_k at state S_k. It is the set of activities that have just finished with the execution. For simplicity, we shall assume that $i_k \neq' next' \Rightarrow F_{k+1} = \varnothing$ and that non-dummy activities have positive durations. The transition function from the current stage k to the next stage $k+1$ is defined as:

$$S^M(S_k, u_k, F_{k+1}) := \begin{cases} t_{k+1} := t_k, A_{k+1} := A_k \cup \{i_k\} \setminus F_{k+1}, & \\ U_{k+1} := U_k \setminus \{i_k\}, \xi^{i_k}_k := t_k, & \\ s_{k+1} := \xi_k, \mathcal{H}_{k+1} := \bar{\mathcal{H}}_k & \text{if } i_k \in U_k \\ t_{k+1} := t_k + 1, A_{k+1} := A_k \setminus F_{k+1}, & \\ U_{k+1} := U_k, s_{k+1} := \xi_k, \mathcal{H}_{k+1} := \bar{\mathcal{H}}_k, & \text{if } i_k =' next' \end{cases} \tag{1}$$

The decision process runs until the terminating state S_L where $U_L = \varnothing$ marks the end of project execution. Transition function can be decomposed as $S^M(S_k, u_k, F_{k+1}) = \sigma(S_k^u = \psi(S_k, u_k), F_{k+1})$ for computational efficiency. Post-decision transition function ψ performs all the parts of calculations for S^M that do not involve F_{k+1}. Stochastic transition function σ performs the remaining calculations using F_{k+1} to complete the transition from post-decision state S_k^u to pre-decision state S_{k+1}.

3.1.4 Cost function

Decisions made at the initial state S_{-1} do not incurr costs. Immediate cost function $g(S_k, u_k)$ denotes the cost of making the decision u_k at the state S_k. It is defined as:

$$g(S_k, u_k) = \sum_{i \in U_k} z_i \left(s_k^i, \xi_k^i, t_k\right) + \mathbf{1}_{i_k = 'next'} \cdot q^+(t_k + 1) + \mathbf{1}_{i_k = n+1} \cdot q^-(t_k) \quad (2)$$

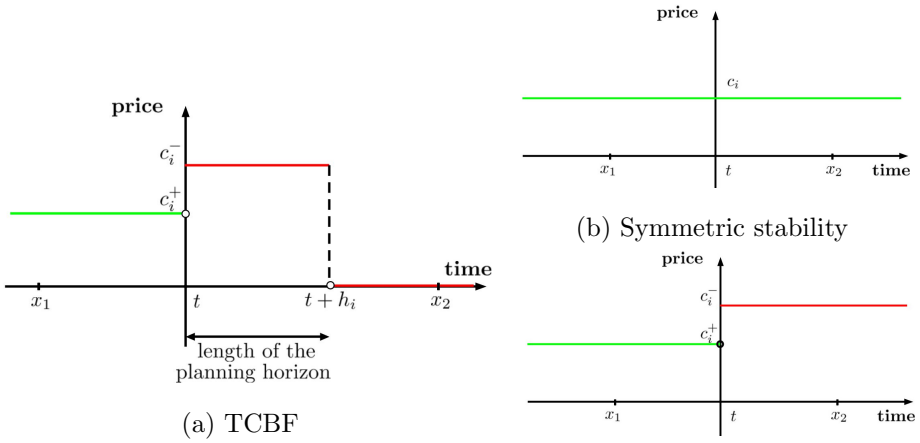
The cost of exceeding the due date is incurred as the stage cost $q^+(t) = \beta^+ \cdot \mathbf{1}_{t>\delta}$ with each time-step. A bonus for completing the project before the due date, $q^-(t) = \beta^- \cdot max(0, \delta - t), \beta^- \leq 0$, is assigned at the end of the project execution. These last two elements are the components of the quality robustness. The first element in immediate cost, the summation of z_i evaluations, is the component of our CBF robustness measure.

The objective function, or **cost-to-go function**, J for policy μ at stage k from state S_k can be written as:

$$J_k^\mu(S_k) = \mathbb{E}[\sum_{j=k}^{L} g(S_k, u_k = \mu(S_k)]). \quad (3)$$

We search for the policy μ^* which minimizes the cost-to-go function. Optimal cost-to-go function J_k^* and policy μ^* can be found using recursive Bellman's equations:

$$\mu^*(S_k) = argmin_{u_k \in \mathcal{X}(S_k)} \mathbb{E}[g(S_k, u_k) + J_{k+1}^*(S^M(S_k, u_k, F_{i+1}))] \quad (4)$$



(a) TCBF

(b) Symmetric stability

(c) Asymmetric stability

Fig. 1. Pricing situation for activity i and two alternative actual baseline times, x_1 and x_2, from the perspective of the current time-point t

3.2 The TCBF measure

As already discussed, the majority of proactive scheduling procedures described in the literature commonly measure solution robustness using the symmetric stability measure (Demeulemeester & Herroelen, 2011, 159). By their definitions, symmetric and asymmetric stability measures do not allow for proactive rescheduling, since they assume at most one change to the baseline time per activity, during the start of activity execution. We can extend their definitions to allow intermediate changes to the baseline. However, even the extended symmetric stability measure introduces a form of inflexibility into the system in such a way that taking advantage of the proactive rescheduling does not have a positive utility (Brčić et al., 2014). Both measures, due to their structure, insist equally on respecting near- and distant-future arrangements by penalizing changes irrespective of their temporal distance.

For these reasons, we introduce a new robustness measure as a component of the objective function J_{-1}, with the form proposed in Brčić et

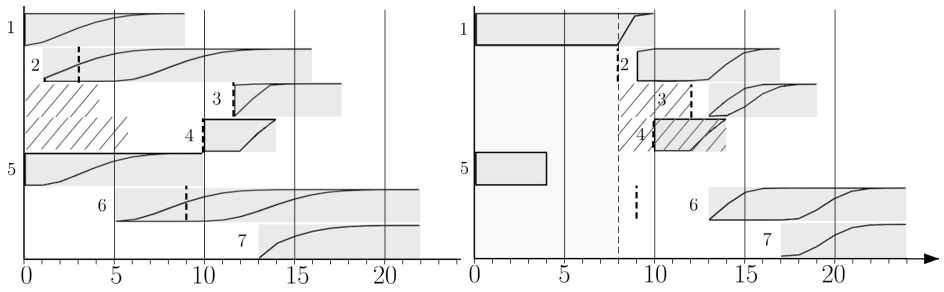
al. (2014). The robustness measure is the TCBF, which is defined by the activity rescheduling cost function $z_i : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{R}$:

$$z_i(x, y, t) := \begin{cases} |y - x| \cdot c_i^+, & \text{if } x < t \\ \min\left\{\max\left\{h_i - (\min\{x, y\} - t), 0\right\}, |y - x|\right\} \cdot c_i^-, & \text{otherwise} \end{cases} \tag{5}$$

where $(\forall i \in V \setminus \{0, n+1\})\ c_i^-, c_i^+ \in \mathbb{R}_+, h_i \in \mathbb{R}_{++} \cup \{\infty\}$. Dummy activities are considered completely flexible, which means that they incur no costs for rescheduling. Note that x is the actual baseline time, y is the new intended baseline time, and t is the time-point at which the update is being made.

Figure 1 compares the symmetric and asymmetric stability measures with the TCBF for one activity. The TCBF is a generalization of asymmetric stability measure from Deblaere et al. (2011), where changes to the elements scheduled in the future are penalized with a threshold.

Such a mechanism captures the idea that for each activity, there is a flexibility in a form of *temporal distance* or *planning horizon*. Executing system can effortlessly or costlessly adapt to any changes beyond the horizon, as the system has not undertaken any committing actions regarding that activity (for example, starting with the preparations). The pricing according to the defined activity rescheduling cost function in (5) is depicted in Fig. 1(a). All the changes to the baseline schedule made at time t (the graph's origin) are penalized depending on the **size of the change** (the distance between the actual and the updated values, dampened by the threshold) and the **temporal distance of the change**. Let us look at the cost calculation according to the TCBF for two cases of rescheduling that are common in the literature. We assume that we start the execution of activity i at the time-point t. This amounts to changing the baseline time for i to $y = t$. If the actual baseline time was $x = x_1$, then the cost of idling (that is, waiting) under the unit price c_i^+ has to be paid for the time interval between x_1 and



(a) Time-point $t = 0$ (b) Time-point $t = 8$

Fig. 2. Stochastic Gantt charts of a situation where TCBF enables proactive change.

t.

This case can be seen in the left half of Fig. 1(a), where the project-execution system already waits for the start of execution according to the actual baseline schedule. On the other hand, if the activity i was originally planned for a future time step $x = x_2$, only the part of the time interval between t and the actual baseline time that intersects the planning horizon is penalized with the unit price c_i^-. This situation can be seen in the right half of Fig. 1(a).

Activity with an actual baseline time in the future can also be rescheduled to any future time-point, without starting the activity. The cost is paid only for the part of the time interval that intersects the planning horizon. This type of change is the basis of proactive rescheduling.

Figure 2 shows a simple situation where TCBF enables proactive change. The project has 6 non-dummy activities. The seventh activity is the finish of the project. Each activity has a horizontal bar that shows the cumulative distribution functions of the start (left-most curve) and end times (right-most curve) for the activity. The left edge of the bar marks the minimum activity start time while the right edge marks the maximum end time. The top edge of the bar marks the cumulative probability of one, and the bottom

edge marks the cumulative probability of zero. Bold dashed lines represent the predicted start times in the baseline schedule. Hatched areas represent planning horizons for inflexible activities that have not yet been started. The activity label is shown to the left of its corresponding bar.

Figure 2(a) shows the stochastic Gantt chart created by the simulation of the RPRT heuristic that incorporated all the available information at the beginning of the project ($t = 0$). The baseline schedule was created so as to minimize the expected deviation costs with respect to the cumulative distribution function (CDF) of start times. At the time-point $t = 8$ we have more information than we had at $t = 0$: activity 5 has finished with the execution at $t = 4$ and activity 1 has still not finished at $t = 8$. The stochastic Gantt chart in Fig. 2(b) has the predictive stochastic part for times $t > 8$ and the realized deterministic part for $t \leq 8$. The previously set baseline times are no longer optimal for activities 2, 3, and 6. Moreover, they are infeasible, since they do not overlap with their corresponding bars. Only activity 3 is inflexible and, ideally, we could change its old baseline time to the new optimum. Such a change will be made only if the sum of the rescheduling cost and the expected cost of the new time is lower than the cost of the old time. As the actual baseline for activity 3 is at the edge of the planning horizon, we shall proactively reschedule its baseline time at no rescheduling cost. In the case of planning horizons of infinite length, the rescheduling cost could offset the benefits of proactive rescheduling.

Davari and Demeulemeester (2017) took a qualitatively different approach and extended symmetric stability measure by adding a fixed penalty for each change to the baseline. Hence, the activities still have infinite planning horizons, but in this case the number of reactions is reduced by batch updates to the baseline schedule. This is valid under assumption that the number of changes is directly related to the performance. However, in this paper we did not make such an assumption.

4 Solution Methods

CBF-SRCPSP, as a generalization of the deterministic RCPSP, is NP-hard. It could be solved exactly by using methods for solving SDPs as explained in Brčić et al. (2014). This was done in (Davari & Demeulemeester, 2017) for a related problem, PR-RCPSP.

However, all exact solving approaches are hit by the three curses of dimensionality: in the state space, in the random information space, and in the control space (Powell, 2011). To deal practically with these problems, we devised metaheuristic algorithms based on approximate dynamic programming that deal with each of the curses by making the following choices:

- state space: we do not calculate the whole cost-to-go vector, as the state space is huge, and we use approximations of the cost-to-go vector;
- random information space: calculations of the expected values are replaced with sampling and simulation using scenarios;
- feasible control space: complex decisions at a single time-point are broken up into a sequence of simple decisions. A search for proactive changes to the baseline takes advantage of the form of the TCBF to restrict the search over feasible control space.

RPRT is fast and high-performing greedy heuristic for which optimal baseline predictions can be efficiently calculated in an external procedure, as demonstrated in Deblaere et al. (2011). We decided to use rollout metaheuristic over RPRT in order to enable proactive rescheduling, hence significantly increasing the dimensionality of decision space. However, that decision space can still be efficiently searched with the help of auxiliary procedure (Section 4.1) that uses the properties of RPRT heuristic and TCBF measure to find the optimal predictive times. All our approaches have offline and online calculation phases. The offline phase ($k = -1$) calculates

the base heuristic and the initial baseline schedule of satisfactory quality using the prior activity-duration distribution. The online phase ($k \geq 0$) collects the information revealed during the project execution to update the activity duration distribution and improve the policy decisions of the base heuristic. The following definitions, adapted from (Goodson et al., 2017) to our problem, are necessary for investigating performance properties.

Definition 1. *Let S be (pre- or post-decision) state in state space $\mathcal{S}$. A* ***heuristic*** *$\mathcal{H}(S)$ is any method to select decision rules in stages $k, k+1, ..., L$. The resulting* ***heuristic policy****, denoted $\mu_{\mathcal{H}(S)} = (\delta_k^{\mu_{\mathcal{H}(S)}}, ..., \delta_L^{\mu_{\mathcal{H}(S)}})$, is the sequence of these decisions rules $\delta_j^{\mu_{\mathcal{H}(S)}}(S_j) : S_j \to \bar{\mathcal{X}}(S_j)$, where $\bar{\mathcal{X}}(S_j) \subseteq \mathcal{X}(S_j)$ is the restricted set of feasible decisions at the stage j.*

Definition 2. *For heuristic $\mathcal{H}(\cdot)$ and rollout policy μ, we say μ is* ***rollout improving*** *if for $k = 0, 1, ..., L$,*

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu}(S_j))|S_k\right] \leq \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k)}}(S_j))|S_k\right]. \tag{6}$$

4.1 Rescheduling the baseline

The auxiliary procedure for efficient proactive rescheduling is used in both of the presented metaheuristic approaches (sections 4.2 and 4.3). It is an adaptation of the work on SBD from Deblaere et al. (2011), which uses the solution of the news-vendor problem for the asymmetric stability measure.

Lemma 1. *The marginal cost function for an activity i in the case of a news-vendor variant that uses the TCBF is*

$$\Delta H(x) = -c_i^+ + (c_i^+ + c_i^-) \cdot P(\hat{s}_i \leq x) - c_i^- \cdot P(\hat{s}_i \leq x - h_i)$$

The proof of Lemma 1 is given in the Appendix.

We have adapted the algorithm to work with the TCBF in two stages. *OptimalTimes* (Algorithm 1) finds the ideal baseline times by ignoring the

previously set baseline times and using Lemma 1. *Adjust* (Algorithm 2) finds the best interpolation between the previous and the new ideal baselines, taking into account the costs of rescheduling the baseline. These two algorithms are always applied in succession.

Algorithm 1 *OptimalTimes*

Input: ST - simulation traces

Output: ξ^* - ideal new baseline

1: **for all** $i \in V$ **do**

2: $P \leftarrow$ CDF of $\hat{s}_i$ from ST_i

3: $\hat{s}_i^{min} \leftarrow \min\{\hat{s}_i \in ST_i\}$

4: $\hat{s}_i^{max} \leftarrow \max\{\hat{s}_i \in ST_i\}$

5: **if** $c_i^+ \neq 0 \vee c_i^- \neq 0$ **then**

6: $\xi_i^* \leftarrow \min\{t \in [\hat{s}_i^{min}, \hat{s}_i^{max}] : P(\hat{s}_i \leq t) \geq \frac{c_i^+}{c_i^+ + c_i^-}\}$

7: $\xi_i^* \leftarrow argmin_{t \in [\xi_i^* - 1, \hat{s}_i^{max})}\{\sum_{j=\xi_i^*-1}^{t} -c_i^+ + (c_i^+ + c_i^-) \cdot P(\hat{s}_i \leq j) - c_i^- \cdot P(\hat{s}_i \leq j - h_i)\} + 1$

8: **else** $\xi_i^* \leftarrow \min\{t \in [\hat{s}_i^{min}, \hat{s}_i^{max}] : P(\hat{s}_i \leq t) \geq \frac{1}{2}\}$

Although the marginal cost function is not necessarily monotonic when using TCBF, and there is no closed-form solution, we get a practical algorithm. OptimalTimes uses simulation traces ST to create P, an empirical cumulative distribution function (CDF) for the start times $\hat{s}_i$ of each activity i (lines 2–4). It uses the median as the predicted time if both the overage and underage costs (c_i^+ and c_i^-, respectively) for an activity are zero (line 8). Otherwise, it performs a linear search over all the time-points that are greater than or equal to the critical percentile for the basic news-vendor problem (lines 6–7). The chosen predicted time is the one with the lowest expected cost, inferred by the summations of the forward differences. The result of this algorithm is the new ideal baseline schedule ξ^*. The baseline ξ^* is optimal at the time-point of calculation under the following assump-

tions: (1) no rescheduling costs are incurred for proactive changes, (2) an unchanged policy that created simulation traces will be used for control, and (3) no future proactive changes will be applied to the baseline. OptimalTimes is used as an approximate algorithm in the proposed controller algorithms because Assumptions (2) and (3) are violated, but that is the price of increased computational efficiency.

Algorithm 2 *Adjust*

Input: ST - simulation traces

Input: s, ξ^* - actual and ideal baseline schedules

Input: t - current time-point

Output: ξ - proposed baseline schedule

1: **for all** $i \in V$ **do**
2: **if** $s_i = \xi_i^*$ **then** $\xi_i \leftarrow s_i$
3: $\rho \leftarrow z_i(s_i, \xi_i^*, t)$
4: **if** $s_i \leq t$ **or** $\xi_i^* < t$ **then**
5: $\xi_i \leftarrow s_i$
6: **else if** $\rho > 0$ **then**
7: $l \leftarrow \min\{s_i, \xi_i^*\}$, $u \leftarrow \max\{s_i, \xi_i^*\}$
8: $\xi_i \leftarrow argmin_{b \in [l,u]}\{z_i(s_i, b, t) + \frac{1}{N}\sum_{j=1}^{N} z_i(b, ST_{i,j}, ST_{i,j})\}$

OptimalTimes, assuming Assumption (1), creates the ideal baseline schedule ξ^*, irrespective of the rescheduling costs from the actual baseline schedule s. In order to fix the violation of that assumption, algorithm Adjust (Algorithm 2) adapts the preliminary solution. It iterates through the activities with proposed changes to the baseline. For each such activity, it calculates the rescheduling cost ρ (line 3). The changes are reset for activities that have their time-points, actual or intended, in the past or present (lines 4–5). In that case, the preparation for activity is done and the system waits for the start of execution. For the rest of the activities that have a

positive rescheduling cost ρ, we perform a linear search over all the time-points between the actual and proposed baseline times to find the optimal interpolated change (lines 6–8).

Algorithm 3 *Flexible Rollout*

Input: $\hbar(S_k)$ - input project at stage $k \geq 0$

1: **while** $U_k \neq \varnothing$ **do**
2: generate N new scenarios $\omega, \forall i \in A_k \cup U_k$
3: **repeat**
4: $\xi_k, \mathcal{F} \leftarrow s_k, Feas(S_k) \cup \{'next'\}$
5: $a, \xi \leftarrow \text{argmin}_{i \in \mathcal{F}}[g(S_k, u) + \mathcal{J}(S_k^u = \psi(S_k, u), \omega) | u = (i, s_k, \mathcal{H})]$
6: **if** $a =' next'$ **then** $\xi_k \leftarrow \xi$ ▷ proactive update
7: $S_{k+1} = S^M(S_k, \delta_k^{\mu_{FR}}(S_k) = (a, \xi_k, \mathcal{H}), F_{k+1})$
8: $k \leftarrow k + 1$
9: **until** $a =' next'$ **or** $U_k = \varnothing$

4.2 Flexible rollout

The flexible rollout (FR) controller (Algorithm 3) uses the RPRT-type base heuristic $\mathcal{H}$ in evaluations during the online computation of control selection. The initial base heuristic and baseline schedule are found by some optimization algorithm (in our experiments SBD) in S_{-1} and they are adapted to all the information known prior to the project start. In each time step, the project state is updated with new information (F_{k+1} in line 7) and the new evaluation scenarios ω are generated according to the updated information (line 2). As a simulation variance-reduction technique, new scenarios reuse scenarios from the previous iteration, updating only the scenario durations of currently active activities. The control selection (line 5) evaluates controls using the set of scenarios and simulation. It is an approximation of (4) that uses reduced decision space and evaluation of the cost-to-go value

of the base heuristic $\mathcal{H}$ instead of the optimal policy. This involves trying out each of the feasible activities as the first step. This results in incurring an immediate cost $g(S_k, u)$ and a transition ψ to the control-specific post-decision state S_k^u. The evaluation of the cost-to-go value of the post-decision state S_k^u is done in $\mathcal{J}$ (Algorithm 4). The baseline is rescheduled only when the control *"next"* is selected (line 6). The controller applies the control specified by the decision rule $\delta_k^{\mu_{FR}}$, which selects a control with the lowest evaluated total cost (line 7).

Algorithm 4 $\mathcal{J}$ *evaluation*

Input: S_k^u - initial post-decision state

Input: ω - N evaluation scenarios

Output: c, ξ - new baseline ξ and its cost c

1: **for all** $i \in [1, N]$ **do**
2: simulate from S_k^u using $\mathcal{H}$ with scenario ω_i
3: $ST_i \leftarrow$ start times in simulated execution
4: $\xi \leftarrow$OptimalTimes(ST)
5: $\xi \leftarrow$Adjust(ST, s_k, ξ, t_k)
6: $c \leftarrow$ average cost of ST under the baseline ξ
7: **return** (c, ξ)

All the scenario runs are completed in a call to $\mathcal{J}$ (Algorithm 4) using the base heuristic $\mathcal{H}$ from S_k^u (lines 1–3). For each post-decision state S_k^u, the optimal baseline schedule over the traces is found **a posteriori**, using algorithms OptimalTimes and Adjust (lines 4–5). The cost-to-go of S_k^u is then evaluated as average cost over the scenarios, according to each S_k^u-optimal baseline and their respective simulation trace data (line 6). Below stated Proposition 1 shows that the performance of the FR is at least as good as RPRT. The proof is given in the Appendix.

Proposition 1. *FR that uses RPRT with finite release times as the base*

heuristic is rollout improving.

4.3 Iterative online simulation-based descend

The iterative online simulation-based descend (IOSBD) controller (Algorithm 5) initially uses the base RPRT heuristic $\mathcal{H}_0$, found by an offline algorithm (in our experiments SBD) in S_{-1}. *Online SBD*(OSBD) is a modification of the SBD procedure from Deblaere et al. (2011) to work online and use our objective function. The former is achieved by omitting the search for an initial policy, reusing the base-heuristic from the previous iteration, and reducing the search over the parameters to only the unstarted activities and only the time-points in the present and the future. The latter is accomplished by using algorithms OptimalTimes and Adjust for determining the optimal predictive starting times.

Algorithm 5 *IOSBD*

Input: $\hbar(S_k)$ - input project at stage $k \geq 0$

1: **while** $U_k \neq \varnothing$ **do**
2: generate N new scenarios $\omega, \forall i \in A_k \cup U_k$
3: $\bar{\mathcal{H}}_k, \xi_k \leftarrow OSBD(S_k, \omega)$ ▷ proactive update
4: **repeat**
5: $a \leftarrow \bar{\mathcal{H}}_k(S_k)$
6: $S_{k+1} = S^M(S_k, \delta_k^{\mu_{IOSBD}}(S_k) = (a, \xi_k, \bar{\mathcal{H}}_k), F_{k+1})$
7: $k \leftarrow k + 1$
8: $\bar{\mathcal{H}}_k, \xi_k \leftarrow \mathcal{H}_k, s_k$
9: **until** $a =' next' \vee U_k = \varnothing$

During the project execution, we use the gradually incoming information (F_{k+1}) to update the project state (line 6). Upon each transition to the next time-point, new uncertainty scenarios ω are generated (line 2). The sequence of base RPRT heuristics $\bar{\mathcal{H}}_k$ is computed by employing the OSBD

procedure (line 3). Scenarios are reused between the stages in order to reduce the simulation variance. The OSBD procedure at the beginning of each time-point returns a new base heuristic $\bar{\mathcal{H}}_k$ and a new optimal baseline proposal ξ_k. During each time-point, base heuristic $\bar{\mathcal{H}}_k$ sequentially selects the activities it can start before making a transition to the next time-point (control *"next"*) (lines 5–8). The controller applies the control specified by the decision rule $\delta_k^{\mu_{IOSBD}}$ which starts the selected activity, and updates the baseline and base heuristic. Below stated Proposition 2 shows that the performance of the IOSBD is at least as good as RPRT. The proof is given in the Appendix.

Proposition 2. *IOSBD is rollout improving over the heuristic that runs the RPRT heuristic stored in the state.*

5 Computational Results

5.1 Experimental setup

We use the performance of RPRT+SBD as the benchmark for comparisons in order to evaluate the effectiveness of the proposed algorithms on projects with the TCBF measure. The RPRT family trained with SBD suffices for the benchmark as it is still, to the best of our knowledge, the best performing proactive–reactive approach. In order to compare our algorithms to the benchmark in a setting as favorable as possible to the RPRT+SBD, we kept our experimental design choices as similar as possible to those found in Deblaere et al. (2011). We used OptimalTimes within the SBD procedure for the optimal baseline schedule calculation in order to adjust for the TCBF measure.

Due to the computational demands of our algorithms, we used only 48, 48, and 60 instances per each of the J30, J60, and J120 instance sets of PSPLIB (Kolisch & Sprecher, 1997), respectively. We used stratified sam-

pling to select one random instance per each of the PSPLIB's parameter groups. The selected instances are listed in the Appendix. Stochasticity in activity durations is modeled identically to Deblaere et al. (2011), using a discretized beta distribution with the shape parameters $\alpha = 2$ and $\beta = 5$ and expected value $\mathbb{E}(d_i)$ set equal to the deterministic activity duration from the PSPLIB instance. Each distribution is scaled with equal probability over the one of $[0.75 \cdot \mathbb{E}(d_i), 1.625 \cdot \mathbb{E}(d_i)]$, $[0.5 \cdot \mathbb{E}(d_i), 2.25 \cdot \mathbb{E}(d_i)]$ or $[0.25 \cdot \mathbb{E}(d_i), 2.875 \cdot \mathbb{E}(d_i)]$ intervals. The experiment is repeated for uniform and discretized Beta(1,3) distributions in the Appendix, with similar results. The parameters for TCBF were generated for each instance in the following way. Every non-dummy activity has a 50% chance to be inflexible—that is, it incurs rescheduling costs. The TCBF costs c_i^+ and c_i^- for inflexible activities are drawn from the discretized triangular distribution with parameters $a = 0, b = 11, c = 0$, and the discretization is done using the ceiling function. The TCBF thresholds for inflexible activities are drawn from the discretized triangular distribution with parameters $a = 0, b = 20, c = 4$. The penalty for exceeding the due date is $\beta_d^+ = 38$. Flexible activities have $c_i^+ = c_i^- = 0$ and the threshold h_i is set to ∞.

From each of the selected J30, J60, and J120 instances, we created **six projects**, according to two factors. The first factor regarding the project early-finish bonus has two levels: no bonus ($\beta_d^- = 0$) and with bonus ($\beta_d^- = -19$). The second factor is the due date tightness, with the following levels: $1.01 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$, $1.05 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$, $1.1 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$, where the expected makespan of the initial policy μ_1 of SBD is used as a basis. We used 1,000 activity-duration scenarios per each created project instance.

RPRT and FR were run on all the instances, but IOSBD was run only on the J30-based instances, due to its long execution times. Sample size used for Monte Carlo simulation in FR and in IOSBD is $N = 1000$, due to the identical sample size used for SBD algorithm in Deblaere et al. (2011).

The results were obtained on virtual machine instances, each having two dedicated logical cores of Intel Xeon processors and 2 GB of RAM with Ubuntu Linux 14.04 and the gcc 4.9.1 C++ compiler (flags -O2 -march=native).

5.2 Performance

We tested all the performance values for statistical significance using multiple paired t-tests at the $p = .05$ level with the Holm-Bonferroni correction, which controls for the family-wise error rate. All the comparisons with RPRT were significant at this level.

For the comparisons between the algorithms, we used the following formula for the calculation of relative improvement of algorithm Alg_2 over Alg_1 on project i:

$$\Delta_i^{Alg_1,Alg_2}(C) = \frac{C_i^{Alg_1} - C_i^{Alg_2}}{|C_i^{Alg_1}|} \cdot 100\% \tag{7}$$

where C is the sample statistic (mean or standard deviation) used for summarizing the algorithm's performance on the project i. When we consider comparisons on a set of projects, we calculate the average of the relative improvements on each project in the set.

Table 1 shows the relative improvements over the experimental factors: project size group, pair of algorithms in comparison, early-completion bonus, and due date tightness. Some of the improvements are greater than 100% because the policy costs can be negative in certain cases due to the early-finish bonus. FR offers a substantial improvement over RPRT in performance mean over projects for all project groups, where the improvements increase with the project size. The comparison of IOSBD with RPRT on the J30-based projects reveals that it seems to be the best-performing algorithm. The first obvious pattern is that projects with a bonus have a higher improvement. The proposed algorithms seem to be even better at reducing the expected makespan when early finishing is rewarded. Finally, projects

Table 1. Relative improvements in the performance mean over experimental factors

	RPRT-FR						RPRT-IOSBD	
	J30		J60		J120		J30	
	N	B	N	B	N	B	N	B
$1.01 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	15.6	69.2	21.7	48.9	31.0	50.3	25.0	108.3
$1.05 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	17.4	230.6	23.0	631.9	29.6	178.2	28.3	384.8
$1.1 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	18.9	41.1	22.8	87.6	27.8	406.2	33.9	67.3

Note. N = No bonus, B = With bonus

with the due date multiplier 1.05 and an early-finish bonus have the greatest improvement. This implies that RPRT substantially underperforms in the situations with a moderately tight due date and an early-finish bonus.

In addition to the mean performance on projects (Table 1), the proposed algorithms also improve the performance variance. FR brings improvements to the standard deviation in comparison to RPRT by 16.4% for J30, 23.6% for J60, and 37.2% for J120 project groups. IOSBD reduces the standard deviation by 18.3% relative to RPRT on J30-based projects.

In Fig. 3, we show the number of proactive updates to the baseline schedule for the algorithms FR and IOSBD in all project instances. The number of proactive updates for the benchmark algorithm RPRT is always zero, since it attempts no such changes. It is evident that IOSBD makes substantially more updates to the baseline schedule than FR on projects of the same size. This is due to a more thorough search over the space of controls through the iterative updates to the base heuristic.

The total cost of a project (3) is the sum of the quality robustness and cost-based flexibility costs. Hence, we can break the total percentage im-

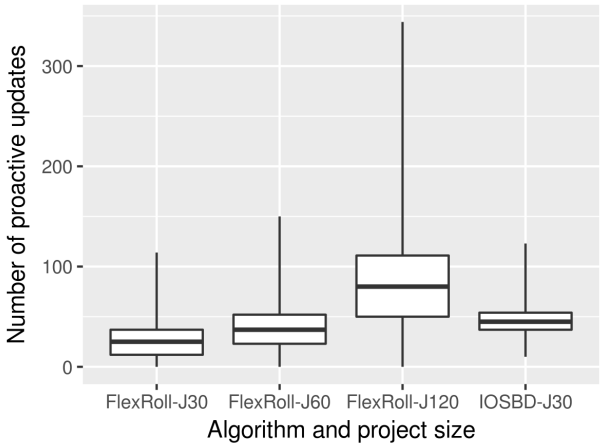


Fig. 3. Number of proactive updates per scenario for FR and IOSBD

provements into the improvements in these two cost components. The relative improvement in cost components is calculated with a modification of (7), where we use component values in the numerator and the corresponding total cost values in the denominator. In Fig. 4, we present scatter plots containing the points that represent the relative improvements in the components for the relevant J30-derived projects. Outliers were removed from all plots to improve the clarity of exposition. Each point in the scatter plot also holds information about a total relative improvement, as a sum of the point's coordinates. Both algorithms, FR and IOSBD, clearly dominate the RPRT, as most of the points in the two left-most scatter plots are located in the first quadrant. In the situations with an early-finish bonus, they tend to improve over the benchmark both in quality and flexibility. In some situations for the case with no bonus, a small loss of performance in quality robustness with respect to RPRT is accepted for a better overall performance. We can see from all three plots that IOSBD does better overall than FR in terms of quality. The rightmost scatter plot also shows that IOSBD is able to trade off flexibility for greater overall improvement in the case of a substantial early-finish bonus. In the absence of the bonus, IOSBD tends

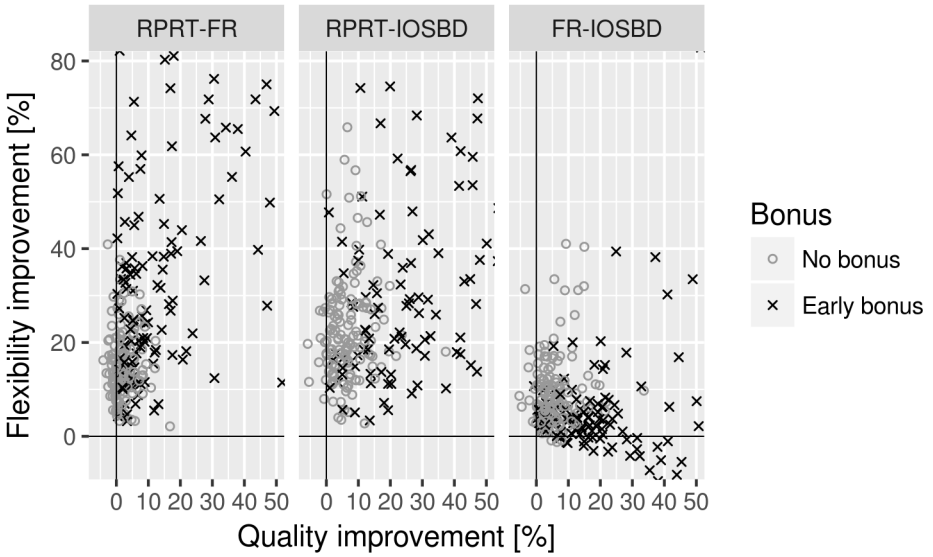


Fig. 4. Relative improvements in the performance components for J30

to shift the improvement focus to the flexibility.

Table 2 shows the average computation times for the algorithms. SBD is the offline algorithm, and the rest are run online. All the online algorithms use SBD for their offline calculations. RPRT is fast and scales well with an increase in project size. FR takes more time but the performance improvements justify the increased computation duration, which is still acceptable. The performance improvements of IOSBD come at the price of a substantial increase in the computation time. Although the computation time per time-point is not excessive, a better optimization algorithm is needed.

6 Conclusion

Real-world projects, in addition to dealing with uncertainties, need to coordinate the actions of project collaborators. Proactive–reactive models make collaborator synchronization possible. However, the current models that use schedule stability measure allow for very little flexibility in adapting

Table 2. Average computation times for algorithms

	J30	J60	J120
SBD	11.2 s	105 s	23 min
RPRT	0.013 ms	0.029 ms	0.073 ms
FR	1.24 s	6.5 s	44.5 s
IOSBD[a]	260.7 s	-	-
FR (p.t.[b])	0.018 s	0.071 s	0.3 s
IOSBD[a] (p.t.[b])	3.68 s	-	-

[a] IOSBD was executed only on J30-derived instances

[b] Per time-point

the baseline schedule to the information made available during the project execution (Brčić et al., 2014). This flexibility deficit makes the proactive rescheduling ineffective for improving the overall performance. We introduced the TCBF measure and the metaheuristic algorithms built on it in order to address these issues.

The TCBF robustness measure relaxes the mentioned rigidity by introducing the notion of planning horizons for activities. Only the part of the change that intersects the activity's planning horizon incurs a rescheduling cost. However, such a setting requires searching in a more complex policy space than do the currently published approaches. This is due to a greatly increased control space, that at each time step enables proactive changes to the whole unrealized part of the baseline schedule.

We proposed the algorithms FlexRollout and IOSBD that function as metaheuristics over the base RPRT heuristic. They deal with the increased policy-space complexity by breaking up the computational burden into offline and non-trivial online parts. We have experimentally demonstrated that our algorithms substantially outperform the best existing method, RPRT

+ SBD from (Deblaere et al., 2011), on models that use TCBF in place of the asymmetric stability measure. We observed simultaneous average improvements in the performance and the standard deviation on the used test sets. The obtained theoretical results guarantee that the performance of our algorithms is, in the worst case, at least as good as that of RPRT.

Presented ideas may be generalized and connected to the general coordination problems in stochastic environments where forecasts are necessary for coordination between the multiple parties. Research into extensions of other robustness measures in a way that would enable proactive rescheduling is a potentially fruitful area. Another direction would be finding more general robustness measures, with weaker assumptions than TCBF, which were made to enable efficient search for the optimal baseline. Rollout metaheuristic with novel, complex decision rules can be used on other methods from the literature to expand decision spaces and efficiently search them to get high-performing closed-loop policies.

References

Artigues, C., Leus, R., & Nobibon, F. T. (2013). Robust optimization for resource-constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, *25*(1), 175–205. doi: 10.1007/s10696-012-9147-2

Bertsekas, D. P., & Castanon, D. A. (1999). Rollout algorithms for stochastic scheduling problems. *Journal of Heuristics*, *5*(1), 89–108. doi: 10.1023/A:1009634810396

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-dynamic programming.* Athena Scientific.

Bertsekas, D. P., Tsitsiklis, J. N., & Wu, C. (1997). Rollout algorithms for combinatorial optimization. *Journal of Heuristics*, *3*(3), 245–262. doi: 10.1023/A:1009635226865

Brčić, M., Kalpić, D., & Katić, M. (2014). Proactive reactive scheduling in resource constrained projects with flexibility and quality robustness requirements. *Combinatorial Optimization*, 112–124. doi: 10.1007/978-3-319-09174-7_10

Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, *18*(3), 263–273. doi: 10.1007/s10951-015-0421-5

Davari, M., & Demeulemeester, E. (2017). The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling*, 1–27. doi: 10.1007/s10951-017-0553-x

Deblaere, F., Demeulemeester, E., & Herroelen, W. (2011). Proactive policies for the stochastic resource-constrained project scheduling problem. *European Journal of Operational Research*, *214*(2), 308–316. doi: 10.1016/j.ejor.2011.04.019

Demeulemeester, E., & Herroelen, W. (2011). *Robust project scheduling.* Now Publishers Inc. doi: 10.1561/0200000021

Goodson, J. C., Thomas, B. W., & Ohlmann, J. W. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, *258*(1), 216–229. doi: 10.1016/j.ejor.2016.09.040

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

Kolisch, R., & Sprecher, A. (1997). PSPLIB - a project scheduling problem library: OR software - ORSEP operations research software exchange program. *European Journal of Operational Research*, *96*(1), 205–216. doi: 10.1016/S0377-2217(96)00170-1

Lambrechts, O. (2007). *Robust project scheduling subject to resource breakdowns* (Unpublished doctoral dissertation). Leuven, Belgium.

Leus, R., & Herroelen, W. (2004). Stability and resource allocation in project planning. *IIE Transactions*, *36*(7), 667–682. doi: 10.1080/07408170490447348

Li, H., & Womer, N. K. (2015). Solving stochastic resource-constrained project scheduling problems by closed-loop approximate dynamic programming. *European Journal of Operational Research*, *246*(1), 20–33. doi: 10.1016/j.ejor.2015.04.015

Möhring, R. H., Radermacher, F. J., & Weiss, G. (1984). Stochastic scheduling problems i — general strategies. *Zeitschrift für Operations Research*, *28*(7), 193–260. doi: 10.1007/BF01919323

Powell, W. B. (2011). *Approximate dynamic programming: Solving the curses of dimensionality, 2nd edition* (2nd ed.). Wiley.

Rostami, S., Creemers, S., & Leus, R. (2016). A new policy class for the stochastic RCPSP. In *Proceedings of the 15th international conference on project management and scheduling* (pp. 102–105).

Sotskov, Y., & Werner, F. (2014). A stability approach to sequencing and scheduling under uncertainty. In Y. Sotskov & F. Werner (Eds.), *Sequencing and scheduling with inaccurate data* (pp. 283–344). Nova Science Publishers, Inc., New York, USA.

Sotskov, Y. N., & Lai, T. C. (2012). Minimizing total weighted flow time under uncertainty using dominance and a stability box. *Computers & Operations Research*, *39*(6), 1271–1289. doi: 10.1016/j.cor.2011.02.001

Stork, F. (2001). *Stochastic resource-constrained project scheduling* (Unpublished doctoral dissertation). Berlin, Germany.

Van de Vonder, S., Demeulemeester, E., Leus, R., & Herroelen, W. (2006). Proactive-reactive project scheduling trade-offs and procedures. In J. Józefowska & J. Weglarz (Eds.), *Perspectives in modern project scheduling* (Vol. 92, pp. 25–51). Springer US. doi: 10.1007/978-0-387-33768-5_2

Planning horizons based proactive rescheduling for stochastic resource-constrained project scheduling problems - Appendix

Mario Brčić[a*], Marija Katić[b], Nikica Hlupić[c]

[a,c]Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia

[b]Birkbeck, University of London, Malet St, Bloomsbury, London WC1E 7HX, United Kingdom

[a]mario.brcic@fer.hr,[b]m.katic@bbk.ac.uk,[c]nikica.hlupic@fer.hr

August 1, 2018

*Corresponding author

1 Sampled projects

As mentioned in the main text, we used 48,48, and 60 randomly selected instances per each of the J30, J60, and J120 instance sets of PSPLIB (Kolisch & Sprecher, 1997), respectively. Below, we list the sampled project instances for each instance set. The names of instances are taken from (Kolisch & Sprecher, 1997). Their form is x_y, where x is the group identifier of parameter values for project generator and y is the identifier of specific generated instance with those parameters. From the names of the instances, it is evident that we have used stratified sampling with the group identifier as strata.

The following projects were sampled from J30: 1_8, 2_4, 3_3, 4_4, 5_2, 6_3, 7_7, 8_3, 9_10, 10_7, 11_5, 12_2, 13_6, 14_2, 15_2, 16_8, 17_10, 18_2, 19_7, 20_10, 21_1, 22_7, 23_10, 24_7, 25_3, 26_9, 27_3, 28_2, 29_9, 30_8, 31_8, 32_5, 33_9, 34_1, 35_5, 36_5, 37_9, 38_4, 39_2, 40_3, 41_6, 42_9, 43_2, 44_9, 45_4, 46_7, 47_1, and 48_2.

The following projects were sampled from J60: 1_2, 2_7, 3_9, 4_5, 5_10, 6_6, 7_7, 8_6, 9_2, 10_5, 11_7, 12_5, 13_9, 14_5, 15_10, 16_3, 17_8, 18_5, 19_7, 20_8, 21_4, 22_2, 23_8, 24_3, 25_4, 26_6, 27_1, 28_4, 29_5, 30_6, 31_7, 32_1, 33_7, 34_9, 35_9, 36_3, 37_3, 38_9, 39_9, 40_2, 41_10, 42_4, 43_9, 44_2, 45_5, 46_2, 47_5, and 48_1.

The following projects were sampled from J120: 1_2, 2_9, 3_10, 4_7, 5_4, 6_9, 7_1, 8_1, 9_2, 10_3, 11_5, 12_4, 13_8, 14_8, 15_6, 16_5, 17_3, 18_7, 19_1, 20_9, 21_4, 22_5, 23_9, 24_1, 25_8, 26_4, 27_10, 28_9, 29_10, 30_6, 31_9, 32_8, 33_1, 34_2, 35_4, 36_3, 37_6, 38_5, 39_4, 40_1, 41_5, 42_7, 43_5, 44_4, 45_10, 46_2, 47_2, 48_1, 49_6, 50_8, 51_1, 52_7, 53_2, 54_2, 55_9, 56_8, 57_3, 58_3, 59_5, and 60_8.

2 Additional experiments

The experiments in the main paper were done on projects with discretized Beta(2,5) distributions modeling uncertain activity durations. In order to strenghten the implications of those results, we repeated the experiments for other probability distributions. Due to the problem setting, we had to use probability distributions with bounded support. We used discrete uniform distributions and discretized Beta(1,3) distributions in two additional experiments. Uniform distribution is symmetric, while Beta(1,3) is more right-skewed than Beta(2,5). Except for the change of used probability distribution family, other experimental settings are identical to the experiment presented in the main paper. The patterns in both experiments are similar to the ones explicated in the main paper. For that reasons, the results are stated as they are without further elaboration, except for the comparison with Beta(2,5) case from the main text.

We shall see that the used probability distribution has an impact on running times of SBD and IOSBD. Also, the probability distribution has noticeable effect on improvements in the cases with early-finish bonus.

2.1 Uniform durations

The range of each distribution was set with equal probability to the one of $[0.75 \cdot \mathbb{E}(d_i), 1.25 \cdot \mathbb{E}(d_i)]$, $[0.5 \cdot \mathbb{E}(d_i), 1.5 \cdot \mathbb{E}(d_i)]$ or $[0.25 \cdot \mathbb{E}(d_i), 1.75 \cdot \mathbb{E}(d_i)]$ intervals, corresponding to the cases of small, medium or large variability.

Table 1 shows improvements for uniform durations. Improvements are greater than in Beta(2,5) case for early-finish bonus with due date multiplier 1.05. On the contrary, for early-finish bonus with due date multiplier 1.1, the improvement seems to be smaller. In addition to the mean performance on projects (Table 1), the proposed algorithms also improve the performance variance. FR brings improvements to the standard deviation in comparison to RPRT by 15.8% for J30, 21.3% for J60, and 37.0% for J120 project

Table 1. Relative improvements in the performance mean over experimental factors for uniform durations

	RPRT-FR						RPRT-IOSBD	
	J30		J60		J120		J30	
	N	B	N	B	N	B	N	B
$1.01 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	14.9	128.3	21.5	162.6	29.5	50.2	22.6	172.9
$1.05 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	15.8	285.7	20.6	773.8	30.8	274.3	24.7	444.1
$1.1 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	16.9	28.4	22.7	33.7	29.7	160.0	30.7	45.0

Note. N = No bonus, B = With bonus

groups. IOSBD reduces the standard deviation by 16.9% relative to RPRT on J30-based projects.

In Table 2 we can see average computation times. SBD and IOSBD on average take less time than in the cases of Beta(2,5) and Beta(1,3). However, Table 1 shows that the improvements for IOSBD over RPRT are similar to those for Beta(2,5). Obviously, probability distribution has an impact on execution durations for SBD and IOSBD, but not so much for other algorithms.

2.2 Beta(1,3) durations

Each distribution is scaled with equal probability over the one of $[0.8 \cdot \mathbb{E}(d_i), 1.6 \cdot \mathbb{E}(d_i)]$, $[0.6 \cdot \mathbb{E}(d_i), 2.2 \cdot \mathbb{E}(d_i)]$ or $[0.4 \cdot \mathbb{E}(d_i), 2.8 \cdot \mathbb{E}(d_i)]$ intervals, corresponding to the cases of small, medium or large variability.

Table 3 shows improvements for Beta(1,3) durations. In addition to the mean performance on projects, the proposed algorithms also improve the performance variance. FR brings improvements to the standard deviation

Table 2. Average computation times for algorithms in the case of uniform durations

	J30	J60	J120
SBD	7.5 s	105.6 s	18.6 min
RPRT	0.014 ms	0.03 ms	0.086 ms
FR	1.4 s	6.0 s	51.2 s
IOSBD[a]	215.0 s	-	-
FR (p.t.[b])	0.021 s	0.069 s	0.37 s
IOSBD[a] (p.t.[b])	3.1 s	-	-

[a] IOSBD was executed only on J30-derived instances

[b] Per time-point

Table 3. Relative improvements in the performance mean over experimental factors for Beta(1,3) durations

	RPRT-FR						RPRT-IOSBD	
	J30		J60		J120		J30	
	N	B	N	B	N	B	N	B
$1.01 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	18.6	80.1	21.4	43.7	31.1	45.9	27.2	121.8
$1.05 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	17.3	487.5	23.2	399.8	30.1	102.8	28.4	749.2
$1.1 \cdot \mathbb{E}_{\mu_1}(s_{n+1})$	19.8	93.3	21.9	107.8	29.4	653.2	32.4	146.5

Note. N = No bonus, B = With bonus

Table 4. Average computation times for algorithms in the case of Beta(1,3) durations

	J30	J60	J120
SBD	16.8 s	42.7 s	20.6 min
RPRT	0.015 ms	0.03 ms	0.083 ms
FR	1.5 s	5.6 s	51.3 s
IOSBD[a]	333.0 s	-	-
FR (p.t.[b])	0.022 s	0.06 s	0.34 s
IOSBD[a] (p.t.[b])	4.6 s	-	-

[a] IOSBD was executed only on J30-derived instances

[b] Per time-point

in comparison to RPRT by 17.3% for J30, 22.0% for J60, and 35.9% for J120 project groups. IOSBD reduces the standard deviation by 18.8% relative to RPRT on J30-based projects.

In Table 4 we can see average computation times. IOSBD on the average takes more time than in the case of Beta(2,5). However, Table 3 shows that the improvements for IOSBD over RPRT are greater than Beta(2,5) case, for most of the factor combinations.

3 Definitions

In this section we give the definitions necessary for the auxiliary lemmas in the following section.

Definition 3. *Heuristic $\mathcal{H}_P(S_k)$ executes the RPRT heuristic $\mathcal{H}_k$ that is part of the state S_k, that is $\mathcal{H}_P(S_k) = \mathcal{H}_k(S_k)$.*

The following definitions and propositions are adapted from (Goodson,

Thomas, & Ohlmann, 2017), where the objective function is maximized, to our problem of minimizing the cost function.

Definition 4. *(Sequentially Improving Heuristics) Let S be a (pre- or post-decision) state in state space $\mathcal{S}$ and let S' be a state such that it is on a sample path induced by policy $\mu_{\mathcal{H}(S)}$. Then, a heuristic $\mathcal{H}(\cdot)$ is sequentially improving if, for all S and subsequent S',*

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta^{\mu_{\mathcal{H}(S)}}(S_j))|S'\right] \geq \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta^{\mu_{\mathcal{H}(S')}}(S_j))|S'\right]. \tag{1}$$

Definition 5. *(Sequentially Consistent Heuristics) Let S be a (pre- or post-decision) state in state space $\mathcal{S}$ and let S' be a state such that it is on a sample path induced by policy $\mu_{\mathcal{H}(S)}$. Then, a heuristic $\mathcal{H}(\cdot)$ is sequentially consistent if for all S and subsequent S',*

$$(\delta_k^{\mu_{\mathcal{H}(S)}}, \delta_{k+1}^{\mu_{\mathcal{H}(S)}}, ..., \delta_L^{\mu_{\mathcal{H}(S)}}) = (\delta_k^{\mu_{\mathcal{H}(S')}}, \delta_{k+1}^{\mu_{\mathcal{H}(S')}}, ..., \delta_L^{\mu_{\mathcal{H}(S')}}). \tag{2}$$

Proposition 3. *(Sequential Consistency Implies Sequential Improvement). If a heuristic is sequentially consistent, then it is also sequentially improving.*

The proof of Proposition 3 is given in (Goodson et al., 2017). It is stated in that paper as Proposition 2.

4 Proofs

4.1 Proof of Lemma 1

Proof. Let $\mathbb{E}[x]^+ = \mathbb{E}[\max(x, 0)]$. Then, the total expected cost for the activity i is

$$H(x) = c_i^+ \cdot \mathbb{E}[\hat{s}_i - x]^+ + c_i^- \cdot \mathbb{E}[x - \hat{s}_i]^+ - c_i^- \cdot \mathbb{E}[x - \hat{s}_i - h_i]^+$$

The leftmost term is due to the realized start time's being greater than the predicted start time, making the left half from Fig. 1 pertinent to the

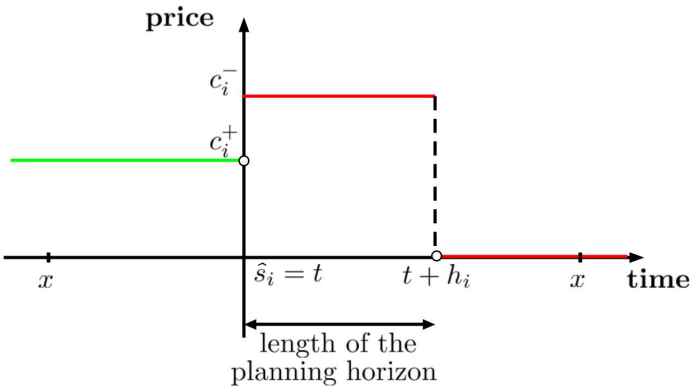


Figure 1. Pricing situation for the two independent cases of actual baseline times, from the perspective of the time-point $\hat{s}_i = t$ at which the activity is started

situation. The two rightmost terms, however, pertain to the right part from Fig. 1, where the cost is incurred only up to the threshold of length h_i. As $x \in \mathbb{N}_0$, let us take the forward difference:

$$\Delta H(x) = H(x+1) - H(x),$$

so we get that

$$\begin{aligned} \Delta H(x) \quad = \quad & c_i^- \cdot \{\mathbb{E}[x+1-\hat{s}_i]^+ - \mathbb{E}\cdot[x-\hat{s}_i]^+\} + c_i^+\{\mathbb{E}[\hat{s}_i - x - 1]^+ - \mathbb{E}[\hat{s}_i - x]^+\} \\ & -c_i^- \cdot \{\mathbb{E}[x+1-\hat{s}_i-h_i]^+ - \mathbb{E}\cdot[x-\hat{s}_i-h_i]^+\} \end{aligned}$$

and using the facts

$$\mathbb{E}[\hat{s}_i - x]^+ = \sum_{j=x}^{\infty} \mathbb{P}(\hat{s}_i > j)$$

$$\mathbb{E}[x - \hat{s}_i]^+ = \sum_{j=0}^{x} \mathbb{P}(\hat{s}_i < j)$$

we get that

$$\Delta H(x) = c_i^- \cdot \mathbb{P}(\hat{s}_i < x+1) - c_i^+ \cdot \mathbb{P}(\hat{s}_i > x) - c_i^- \cdot \mathbb{P}(\hat{s}_i < x+1-h_i)$$

Tidying up, we get

$$\Delta H(x) = c_i^- \cdot \mathbb{P}(\hat{s}_i \leq x) - c_i^+ \cdot [1 - \mathbb{P}(\hat{s}_i \leq x)] - c_i^- \cdot \mathbb{P}(\hat{s}_i \leq x - h_i)$$

from which the claim follows. □

4.2 Proofs of auxiliary lemmas

In this subsection we state and prove lemmas used for the proofs of Proposition 1 and Proposition 2.

Lemma 2. *RPRT heuristic (π, τ) is terminating if and only has finite release times τ.*

Proof. Assuming all the release times are finite, there is the maximal release time τ_{max}. At the time-points $t \geq \tau_{max}$ RPRT heuristic is identical to the parallel schedule generation scheme (PSGS) that uses the activity priority list π. Since PSGS is terminating, so is the RPRT.

If the RPRT heuristic is terminating, then all activities must finish in finite time t_L. By definition, this means that all the release times are less than or equal to t_L. □

Lemma 3. *Terminating RPRT heuristic is sequentially consistent.*

Proof. Assume that RPRT heuristic generates a feasible sequence of activity-starting decisions $([i,t],[i_1,t_1],...,[i_{n+1},t_{n+1}])$, starting at $[i,t]$ where $i \in V$ and t is the time of starting activity i, such that $j < l \Rightarrow t_j \leq t_l$. Also assume that it does not generate the feasible sequence $([i_1,t_1],...,[i_{n+1},t_{n+1}])$ starting at $[i_1,t_1]$. Since the priority values of all activities stay the same, as well as the release times, this can happen only when the sequence $([i_1,t_1],...,[i_{n+1},t_{n+1}])$ is time-, resource-infeasible or release-forbidden, a contradiction. □

Lemma 4. *SBD produces terminating RPRT heuristic in a finite time.*

Proof. SBD procedure is run in the initial state S_{-1}, at the stage $k = -1$.

SBD carries out an improvement on the input RPRT heuristic, $\mathcal{H}_{input} = \Pi_0$. Π_0 is a RPRT heuristic with finite release times which is created from a deterministic schedule. SBD changes the current RPRT heuristic if and only if the expected cost of the new RPRT heuristic, combined with the new baseline schedule, is lower. Non-terminating RPRT have non-finite release time for at least one activity and hence have infinite cost due to the quality robustness that penalizes the amount by which the due date was exceeded.

SBD is greedy descent methods that searches through a finite search space for non-dummy unstarted activities. The number of possible vectors of priority values is finite as only the permutations of a limited number of elements are considered. The release times of unstarted non-dummy activities are limited by the cost of the input heuristic $\mathcal{H}_{input}$ in such a way that $(\forall i \in U_k \setminus \{0, n+1\}) \max\{t_k, 0\} \leq \tau_i \leq \frac{J_k^{\mu_{\mathcal{H}_{input}}(S_k)}(S_k)}{\beta^+} + \max\{t_k, \delta\}$. The potential solution is updated only if better solution has been found in the finite search space. Also, SBD terminates if no improvement has been found during one pass through the improvement loop. Hence, SBD produces a solution in finite time.

□

The following corollary follows from Lemma 4. Its proof is a straightforward extension of the proof for Lemma 4.

Corollary 1. *OSBD produces terminating RPRT heuristic in finite time if the input RPRT heuristic is terminating.*

Lemma 5. *FR, that uses terminating RPRT as a base heuristic, is terminating.*

Proof. We shall use the proof by contradiction. Let us take any CBF-SRCPSP $\hbar$, any terminating RPRT $\mathcal{H} = (\pi, \tau)$ for $\hbar$ as the base heuristic, and assume that FR is not terminating.

There has to be at least one activity duration scenario under which FR does not terminate. We examine the behavior of FR under any such scenario ω_{NT}. The project $\hbar$ has finite number of activities and all the activities have bounded durations. Hence, there has to be the minimal timepoint t_b such that all of the following holds:

1. $t_b \geq \delta \wedge t_b \geq \max_{i \in V \setminus \{0,n+1\}} \tau_i$,
2. no activity is started at any timepoint $t \geq t_b$ for the scenario ω_{NT},
3. all the resources are free,
4. the set of unstarted activities is non-empty.

Base heuristic $\mathcal{H}$, when run at timepoints $t \geq t_b$, behaves like the PSGS policy with activity priority vector π.

PSGS policies are time-invariant, since their decisions on starting activities do not depend on the current time, but only on: the activity priority vector π, the set of active, and the set of unstarted activities. For some activity duration scenario η, let us assume that we execute project in any state $S_k = \{t_k, A_k, U_k, s_k, \mathcal{H}_k\}$ with all the project resources free and nonempty set of unstarted activities U_k. Let us note with $j_{t_p}^{f,\eta}(S_k)$ start times of activities in U_k for scenario η if starting no activities for $t < t_p$ and executing PSGS heuristic f for $t \geq t_p$. Then, $(\forall a \in U_k)(\forall z \in \mathbb{N}_0)(\forall t \geq t_k) j_{t+z,a}^{f,\eta}(S_k) = j_{t,a}^{f,\eta}(S_k) + z$ which we shall refer to as the **translational property** of PSGS policies.

Each baseline s_k was generated in a sequence of timesteps, resulting in baselines $s_0, ..., s_k$. Baseline s_0 is finite since it was created using algorithm OptimalTimes in the offline phase (for example, as a part of SBD offline calculation) from simulation traces of the terminating $\mathcal{H}$. At each stage $k > 0$, the baseline s_k was calculated as an interpolation (by Adjust) between the previous baseline s_{k-1} and the new ideal baseline ξ_{k-1}^* (calculated by

OptimalTimes from the simulation traces of terminating $\mathcal{H}$). Since s_k is an interpolation between the two finite vectors, it is also finite.

For stages k where $t \geq t_b$, ideal baseline times ξ^*_k are translated with t (due to the translation property), as they are calculated from the translating simulation traces of $\mathcal{H}$ which at $t \geq t_b$ behaves as PSGS. The actual baselines s_k are interpolations between the previous baseline s_{k-1} and the ideal baseline ξ^*_{k-1}. Therefore, there exist stage $\hat{k}$ and state $S_{\hat{k}} = (\hat{t}, A_{\hat{k}}, U_{\hat{k}}, s_{\hat{k}})$ with the minimal time $\hat{t}$ such that $\hat{t} \geq t_b$ and $(\forall a \in V) s_{\hat{k}} \leq \xi^*_{\hat{k}}$. $\mathcal{H}(S_{\hat{k}})$ starting from $S_{\hat{k}}$ immediately starts the activities $a_1, ..., a_n$ at $\hat{t}$ according to the priority list π.

Let us compare two options for the immediate decision that FR takes into account during the evaluation at state $S_{\hat{k}}$:

1. select control 'next' to change the baseline to $\xi_{\hat{k}}$ and jump to the timepoint $\hat{t} + 1$

2. start the activity a_1.

Base heuristic $\mathcal{H}$ for each simulation scenario, due to the translational property during the evaluation of controls 'next' and a_1, starts activities one timestep later in the option 1 than under the option 2. Let us for simulation scenario η denote r^η_a start time of activity $a \in U_{\hat{k}}$ if we execute heuristic policy $\mu_{\mathcal{H}(S_{\hat{k}})}$ from the state $S_{\hat{k}}$. Then $\forall a \in U_{\hat{k}}$, in option 1, start times under the scenario η are $r'^\eta_a := r^\eta_a + 1$.

Also, let $\xi'^{next'}_{\hat{k}}$ be the proposed baseline schedule for the control 'next' (calculated in $\mathcal{J}$). Let $\mathcal{B} \subseteq U_{\hat{k}}$ be the maximal set of activities for which $(\forall i \in \mathcal{B}) \xi'^{next'}_{\hat{k},i} > s_{\hat{k},i}$. Let us construct $\xi^{a_1}_{\hat{k}}$ in the following way: $\xi^{a_1}_{\hat{k},i} := \xi'^{next'}_{\hat{k},i} - 1$ for all $i \in \mathcal{B}$, and $\xi^{a_1}_{\hat{k},i} := \xi'^{next'}_{\hat{k},i}$ otherwise.

The expected cost of option 1 is calculated by FR using the set of simulation scenarios H in line 6 is:

$$\beta^+ + \sum_{i\in\mathcal{B}} z_i\left(s^i_{\hat{k}}, \xi'^{next'}_{\hat{k},i}, \hat{t}\right) + \frac{1}{|H|}\sum_{\eta\in H}\left[\sum_{t=\hat{t}+1}^{r'^{\eta}_{n+1}} \beta^+ + \sum_{i\in U_{\hat{k}}} z_i\left(\xi'^{next'}_{\hat{k},i}, r'^{\eta}_i, r'^{\eta}_i\right)\right]. \tag{3}$$

The expected cost of option 2, with the baseline $\xi^{a_1}_{\hat{k}}$ is:

$$\sum_{i\in\mathcal{B}} z_i\left(s^i_{\hat{k}}, \xi^{a_1}_{\hat{k},i}, \hat{t}\right) + \frac{1}{|H|}\sum_{\eta\in H}\left[\sum_{t=\hat{t}}^{r'^{\eta}_{n+1}-1} \beta^+ + \sum_{i\in U_{\hat{k}}} z_i\left(\xi^{a_1}_{\hat{k},i}, r'^{\eta}_i - 1, r'^{\eta}_i - 1\right)\right]. \tag{4}$$

It is evident that all the coresponding summands in Equation (4) are less than or equal to the corresponding ones in Equation (3). Due to the additional positive term β^+, Equation (3) is strictly greater. Evaluation of option 2 in line 6 of FR is less than or equal to Equation (4). Therefore, the control 'next' would not be selected in FR at the stage $\hat{k}$, which is a contradiction.

□

Lemma 6. *RPRT heuristic with finite release times is sequentially improving.*

Proof. RPRT with finite release times is terminating (Lemma 2) and sequentially consistent (Lemma 3). Hence, by Proposition 3, it is sequentially improving. □

Lemma 7. *For any state $S_k \in \mathcal{S}$ it holds that*

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}^P(S_k)}}(S_j))\middle| S_k\right] \tag{5}$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}^P(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{6}$$

Proof. Let us mark by $\gamma_k^{\mu}(S_k)$ the **activity starting rule** that decides the activity to be started under policy μ at state S_k.

Under $\mathcal{H}_P(\cdot)$, the base heuristic $\mathcal{H}_k$ remains unchanged for all the subsequent states, that is $\forall j \in [k+1, L]\mathcal{H}_j = \mathcal{H}_k$. This is due to the fact that $\mathcal{H}_P(\cdot)$ executes RPRT heuristic which does not change the baseline schedule nor the heuristic stored in the state. Instead, RPRT heuristics propagate the both to the following stage.

The definition of decision rule $\delta_k^{\mu_{IOSBD}}(S_k)$ depends on the state S_k. We separately prove the claim for each of the two cases.

1. If in S_k some activity was already started at t_k, then $\delta_k^{\mu_{IOSBD}}(S_k) = \delta_k^{\mu_{\mathcal{H}_P(S_k)}}(S_k)$.

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_k)}}(S_j))\middle| S_k\right]$$

$$= \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_k(S_k)}}(S_j))\middle| S_k\right] \tag{7}$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{\mathcal{H}_k(S_k)}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_k(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{8}$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{\mathcal{H}_k(S_k)}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_{k+1}(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{9}$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{\mathcal{H}_P(S_k)}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{10}$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{11}$$

The equality in Equation (7) results from the definition of heuristic $\mathcal{H}_P(\cdot)$. Equation (8) follows from the sequential improvement of RPRT heuristic $\mathcal{H}_k(\cdot)$. Equation (9) follows from fact that $\mathcal{H}_k(S_{k+1}) = \mathcal{H}_{k+1}(S_{k+1})$. The equality in Equation (10) results from the definition of heuristic $\mathcal{H}_P(\cdot)$. Finally, Equation (11) is due to the definition of decision rule for IOSBD in this case.

2. Otherwise, OSBD search procedure is run at the state S_k. It takes the RPRT heuristic $\mathcal{H}_k$, as an input (as the part of the state S_k). OSBD outputs RPRT heuristic $\bar{\mathcal{H}}_k$ and a new baseline schedule ξ_k. Then, the decision rule is $\delta_k^{\mu_{IOSBD}}(S_k) = (\gamma^{\mu_{\bar{\mathcal{H}}_k(S_k)}}(S_k), \xi_k, \bar{\mathcal{H}}_k)$. By the construction of OSBD algorithm and its objective function, $\bar{\mathcal{H}}_k$ and ξ_k are such that the following holds:

$$\begin{aligned} &\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta^{\mathcal{H}_k(S_k)}(S_j))|S_k\right] \geq \\ &\mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k) + \sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\bar{\mathcal{H}}_k(S_k)}}(S_j))|S_k\right]. \end{aligned} \tag{12}$$

Hence, we get:

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_k)}}(S_j))\middle| S_k\right]$$

$$= \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta^{\mathcal{H}_k(S_k)}(S_j))|S_k\right] \tag{13}$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k) + \sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\bar{\mathcal{H}}_k(S_k)}}(S_j))|S_k\right] \tag{14}$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\bar{\mathcal{H}}_k(S_{k+1})}}(S_j))|S_{k+1}\right]|S_k\right] \tag{15}$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_{k+1}(S_{k+1})}}(S_j))|S_{k+1}\right]|S_k\right] \tag{16}$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_{k+1})}}(S_j))|S_{k+1}\right]|S_k\right] \tag{17}$$

The equality in Equation (13) results from the definition of $\mathcal{H}_P(\cdot)$. Equation (14) follows from the operation of OSBD algorithm and Equation (12). Equation (15) is due to the sequential improvement property of RPRT heuristic $\bar{\mathcal{H}}_k(\cdot)$ and iterated expectations. The equality in Equation (16) results from the fact that $\mathcal{H}_{k+1} = \bar{\mathcal{H}}_k$ due to the decision rule of IOSBD in this case. The equality in Equation (17) is because of the definition of $\mathcal{H}_P(\cdot)$.

□

Lemma 8. *IOSBD is terminating for terminating initial base heuristic* $\mathcal{H}_0$.

Proof. Proof by contradiction. Let us assume that IOSBD is not terminating for some project $\hbar$, terminating initial RPRT base heuristic $\mathcal{H}_0$, and scenario ω. From Corollary 1, it follows that $\forall k > 0, \mathcal{H}_k$ is terminating. As the project $\hbar$ has finite number of activities, there has to be a minimal time t_b such that no activity is started at any timepoint $t \geq t_b$ and where all the resources are free and the set of unstarted activities is non-empty. Let $\hat{k}$ be the stage at which t_b is reached under the scenario ω and $\mathcal{H}_{\hat{k}}$ is the base-heuristic at that state. Let $\tau^{\hat{k}}$ be the minimum release time in $\mathcal{H}_{\hat{k}}$ for unstarted resource- and precedence-feasible activities. In order not to start any activity, IOSBD procedure must increase the minimum release time $\tau^{\hat{k}}$ in timepoints $t \in [t_b, \tau^{\hat{k}}]$, or an activity will be started. However, $\mathcal{H}_{\hat{k}}$ is the local optimum of the OSBD procedure for the stage $\hat{k}$ under the set of activity-duration scenarios drawn from $p(d|S_{\hat{k}})$. At each stage k_m taking place at $m \in [t_b, \tau^{\hat{k}}]$, we have $p(d|S_{k_m}) = p(d|S_{\hat{k}})$. Moreover, IOSBD reuses scenarios over stages, which means that the identical set of simulation scenarios is used for all the stages k_m. Hence, OSBD procedure will not change $\mathcal{H}_{\hat{k}}$ until $\tau^{\hat{k}}$, which is a contradiction.

□

Lemma 9. *For all x_1, x_2, x_3 greater than or equal to t it holds:*

$$(\forall i \in V) z_i(x_1, x_2, t) + z_i(x_2, x_3, t) \geq z_i(x_1, x_3, t)$$

Proof. For brevity, when defining intervals we assume that the limits of intervals are reordered.

Obviously, $[x_1, x_2] \cup [x_2, x_3] = [x_1, x_3]$. Let $[x, y] = [x_1, x_2] \cap [x_2, x_3]$ be

the overlap of the two ranges.

$$
\begin{aligned}
& z_i(x_1, x_2, t) + z_i(x_2, x_3, t) \\
&= \sum_{j=\min\{x_1,x_2\}+1}^{\max\{x_1,x_2\}} c_i^- \cdot \mathbf{1}_{j \le t+h_i} + \sum_{j=\min\{x_2,x_3\}+1}^{\max\{x_2,x_3\}} c_i^- \cdot \mathbf{1}_{j \le t+h_i} && (18) \\
&= \sum_{j=\min\{x_1,x_3\}+1}^{\max\{x_1,x_3\}} c_i^- \cdot \mathbf{1}_{j \le t+h_i} + \sum_{j=\min\{x,y\}+1}^{\max\{x,y\}} c_i^- \cdot \mathbf{1}_{j \le t+h_i} && (19) \\
&\ge \sum_{j=\min\{x_1,x_3\}+1}^{\max\{x_1,x_3\}} c_i^- \cdot \mathbf{1}_{j \le t+h_i} && (20) \\
&= z_i(x_1, x_3, t) && (21)
\end{aligned}
$$

Equality in Equation (18) results from the definition of TCBF. Equation (20) is the result of regrouping of the summands. Equation (20) follows from the omission of the non-negative cost. Equality in Equation (21) results from the definition of TCBF.

□

4.3 Proof of Proposition 1

Proof. Proof by induction. FR is terminating (Lemma 5). The result holds trivially for the terminal $j = L$. We assume that the result holds for $j = k+1, ..., L-1$.

Let $\hat{a}_k = \delta_k^{\mathcal{H}(S_k)}(S_k)$.

For the case $j = k$:

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k)}}(S_j))\middle| S_k\right]$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{\mathcal{H}(S_k)}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{\hat{a}_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{22}$$

$$\geq \min_{a \in \mathfrak{X}(S_k)|\mathcal{H}} \mathbb{E}\left[g(S_k, a = (i_k, \xi_k, \mathcal{H})) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^a)}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{23}$$

$$= \mathbb{E}\left[g(S_k, \tilde{a}_k = (\tilde{i}_k, \tilde{\xi}_k, \mathcal{H})) + \mathbb{E}\left[g(S_{k+1}, \delta_{k+1}^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \tag{24}$$

Equation (22) follows from the fact that RPRT base heuristic with finite release times is sequentially improving (Lemma 6) and using iterated expectations. Equation (23) follows from the minimization. Let $\tilde{a}_k = (\tilde{i}_k, \tilde{\xi}_k, \mathcal{H})$ be the optimum of the Equation (23). The equality in Equation (24) follows from our definition of optimum in Equation (23).

There are two cases relevant for the rest of the proof:

1. $\tilde{i}_k =' next'$

   In this case, we use the induction hypothesis to finalize the proof.

2. $\tilde{i}_k \neq' next'$

   In this case, we repetitively expand the cost expression to the following stages until we find the minimal stage $l \geq k+1$ for which the case 1 is true, or $i_l = n+1$ so we are at the terminal case $l = L$. The number of stages necessary for finding l is limited by the number of unstarted activities in the following way $l - k < |U_k|$.

If $\tilde{i}_k =' next'$, then $\delta_k^{\mu_{FR}}(S_k) = \tilde{a}_k$ and the following follows from the Equation (24):

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{FR}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \quad (25)$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{FR}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \quad (26)$$

$$\geq \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{FR}}(S_j))\middle| S_k\right] \quad (27)$$

The equality in Equation (25) results from the definition of FR. Equation (26) follows from the fact that the used heuristic $\mathcal{H}$ is sequentially improving. Equation (27) follows from the induction hypothesis.

If $\tilde{i}_k \neq' next'$, then the baseline schedule is not updated and $\delta_k^{\mu_{FR}}(S_k) = (\tilde{i}_k, s_k, \mathcal{H})$. When $\tilde{i}_k = n+1$ then there are no proactive changes to the baseline and by definition $k = L$, hence the claim holds. Otherwise, we shall postpone the proactive change of the baseline to the next stage, $k+1$. Let us mark by $\gamma_k^{\mu}(S_k)$ the **activity starting rule** that decides the activity to be started under policy μ at state S_k. Let $\delta_{k+1}^{\mu_{\mathcal{H}(S_k^a),\tilde{\xi}_k}}(S_{k+1}) = (\gamma_{k+1}^{\mu_{\mathcal{H}(S_k^a)}}(S_{k+1}), \tilde{\xi}_k, \mathcal{H})$. It starts an activity $i_{\mathcal{H}} \in U_{k+1} \cup \{'next'\}$ at t_k according to the RPRT heuristic $\mathcal{H}(S_k^a)$, and changes the baseline schedule to $\tilde{\xi}_k$ for all the activities except for $i_{\mathcal{H}}$. The next follows from the Equation (24):

$$\mathbb{E}\left[g(S_k, \tilde{a}_k) + \mathbb{E}\left[g(S_{k+1}, \delta_{k+1}^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right]$$

$$\geq \mathbb{E}\left[g(S_k, a_k = (\tilde{i}_k, s_k, \mathcal{H})) + \mathbb{E}\left[g(S_{k+1}, \delta_{k+1}^{\mu_{\mathcal{H}(S_k^{a_k}),\tilde{\xi}_k}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{a_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \quad (28)$$

$$= \mathbb{E}\left[g(S_k, \delta_k^{\mu_{FR}}(S_k)) + \mathbb{E}\left[g(S_{k+1}, \delta_j^{\mu_{\mathcal{H}(S_k^{a_k}),\tilde{\xi}_k}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{a_k})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \quad (29)$$

RPRT heuristics are independent of the baseline schedule so $\mathcal{H}(S_k^{a_k}) = \mathcal{H}(S_k^{\tilde{a}_k})$. Activity durations are also independent of the baseline schedule. Let us mark by $S_{k+1}^{(24)}$ and $S_{k+1}^{(28)}$ states at stage $k+1$ in Equations (24) and (28), respectively. As the set F_{k+1} of finished activities between the stages k and $k+1$ is identical for both the equations, $S_{k+1}^{(24)}$ and $S_{k+1}^{(28)}$ differ only in the baseline schedule. Activity starting decisions of RPRT heuristics do not depend on baseline schedule, hence $\gamma_{k+1}^{\mu_{\mathcal{H}(S_k^{\tilde{a}_k})}}(S_{k+1}^{(24)}) = \gamma_{k+1}^{\mu_{\mathcal{H}(S_k^{a_k}),\tilde{\xi}_k}}(S_{k+1}^{(28)})$. As F_{k+2} is identical for both transitions to the stage $k+2$, it follows that at the stage $k+2$, we end up in the states $S_{k+2}^{(24)} = S_{k+2}^{(28)}$. Therefore, the sums of costs in stages $j \geq k+2$ are identical in Equations (24) and (28).

Both the states S_k and S_{k+1} occur at the identical time-point t_k. TCBF cost due to the postponed rescheduling of the baseline in Equation (28) is identical for all the activities with the exception of $i_{\mathcal{H}}$ (if $i_{\mathcal{H}} \neq' next'$). If $i_{\mathcal{H}} \neq' next'$ then, by postponing the update, we omit one reschedule for $i_{\mathcal{H}}$ and Equation (28) follows from Lemma 9. Equation (29) follows from the definition of FR.

For simplicity, we continue to work just with the inner expectation in (29):

$$\mathbb{E}\left[g(S_{k+1}, \delta_j^{\mu_{\mathcal{H}(S_k^{a_k}),\tilde{\xi}_k}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k^{a_k})}}(S_j))\middle| S_{k+1}\right] \tag{30}$$

$$= \mathbb{E}\left[g(S_{k+1}, \delta_j^{\mu_{\mathcal{H}(S_{k+1}),\tilde{\xi}_k}}(S_{k+1})) + \sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_{k+1})}}(S_j))\middle| S_{k+1}\right] \tag{31}$$

$$\geq \mathbb{E}\left[g(S_{k+1}, a = \delta_j^{\mu_{\mathcal{H}(S_{k+1}),\tilde{\xi}_k}}(S_{k+1})) + \mathbb{E}\left[\sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_{k+1}^a)}}(S_j))\middle| S_{k+2}\right]\middle| S_{k+1}\right] \tag{32}$$

$$\geq \min_{a \in \mathcal{X}(S_{k+1})|\mathcal{H}} \mathbb{E}\left[g(S_{k+1}, a = (i_{k+1}, \xi_{k+1}, \mathcal{H})) + \mathbb{E}\left[\sum_{j=k+2}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_{k+1}^a)}}(S_j))\middle| S_{k+2}\right]\middle| S_{k+1}\right] \tag{33}$$

The cost of changing the baseline to $\tilde{\xi}_k$ in Equations (30) and (31) is

identical and the rest of the costs depends only on the heuristic $\mathcal{H}$. Equality in Equation (31) follows from the sequential consistency of $\mathcal{H}(\cdot)$, since the identical starting times of activities incur identical costs. Equation (32) follows from the sequential improvement of the used RPRT heuristic and iterated expectations. Equation (33) follows from the minimization.

The Equation (33) is of the identical form as the Equation (23), with the stage number increased. The procedure from Equations (28)-(33) can be repeated finite number of times (limited by the number of unstarted activities) until we find the minimal stage number $l \geq k+1$ such that $\tilde{i}_l =' next'$ or $\tilde{i}_l = n+1$. If $\tilde{i}_l = n+1$, then we can use the terminal case and the claim holds. If $\tilde{i}_l =' next'$ we can proceed by using Equations (25)-(27) with $k = l$ to get the following:

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_k)}}(S_j))\middle| S_k\right] \tag{34}$$

$$\geq \mathbb{E}\left[\sum_{j=k}^{l} g(S_j, \delta_j^{\mu_{FR}}(S_j)) + \mathbb{E}\left[\sum_{j=l+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}(S_{l+1})}}(S_j))\middle| S_{l+1}\right]\middle| S_k\right] \tag{35}$$

$$\geq \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{FR}}(S_j))\middle| S_k\right] \tag{36}$$

Equation (35) follows from the application of procedure from Equations (28)-(33) for stages $j < l$ followed by using Equations (25)-(26) for the stage l. Equation (36) follows from the induction hypothesis. This completes the proof.

□

4.4 Proof of Proposition 2

Proof. We use proof by induction. IOSBD is terminating (Lemma 8). The result holds trivially for the terminal $j = L$ case by using Lemma 7.

We assume that the result holds for $j = k+1, ..., L-1$.

For the case $j = k$:

$$\mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_k)}}(S_j))\middle| S_k\right]$$

$$\geq \mathbb{E}\left[g(S_k, \delta_k^{\mu_{IOSBD}}(S_k)) + \mathbb{E}\left[\sum_{j=k+1}^{L} g(S_j, \delta_j^{\mu_{\mathcal{H}_P(S_{k+1})}}(S_j))\middle| S_{k+1}\right]\middle| S_k\right] \quad (37)$$

$$\geq \mathbb{E}\left[\sum_{j=k}^{L} g(S_j, \delta_j^{\mu_{IOSBD}}(S_j))\middle| S_k\right] \quad (38)$$

Equation (37) results from Lemma 7. Equation (38) follows from the induction hypothesis. □

References

Goodson, J. C., Thomas, B. W., & Ohlmann, J. W. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, *258*(1), 216–229. doi: 10.1016/j.ejor.2016.09.040

Kolisch, R., & Sprecher, A. (1997). PSPLIB - a project scheduling problem library: OR software - ORSEP operations research software exchange program. *European Journal of Operational Research*, *96*(1), 205–216. doi: 10.1016/S0377-2217(96)00170-1