Minimizing the expected makespan of a project with stochastic activity durations under resource constraints

Stefan Creemers

IESEG School of Management
s.creemers@ieseg.fr

KU Leuven
stefan.creemers@kuleuven.be

Abstract

The resource-constrained project scheduling problem (RCPSP) has been widely studied. A fundamental assumption of the basic type of RCPSP is that activity durations are deterministic (i.e., they are known in advance). In reality, however, this is almost never the case. In this article we illustrate why it is important to incorporate activity duration uncertainty, and develop an exact procedure to optimally solve the stochastic resource-constrained scheduling problem (SRCPSP). A computational experiment shows that our approach works best when solving small-to medium-sized problem instances where activity durations have a moderate-to-high level of variability. For this setting, our model outperforms the existing state-of-the-art. In addition, we use our model to assess the optimality gap of existing heuristic approaches, and investigate the impact of making scheduling decisions also during the execution of an activity rather than only at the end of an activity.

1 Introduction

The resource-constrained project scheduling problem (RCPSP) is one of the most widely studied scheduling problems. The basic type of RCPSP is to find a precedence- and resource-feasible schedule (i.e., a vector of activity starting times) that minimizes the makespan of a project. One of the fundamental assumptions of the basic type of RCPSP is that activity durations are deterministic (i.e., they are known in advance). In reality, however, activity durations are subject to considerable uncertainty. The stochastic RCPSP (or SRCPSP) takes this uncertainty into account and observes the RCPSP when activity durations are stochastic. In contrast to the basic type of RCPSP, the SRCPSP has received only little attention in the literature (refer to Demeulemeester and Herroelen (2002) and Neumann et al. (2003) for an overview of the field of resource-constrained project scheduling, and to Herroelen and Leus (2005) for a survey on project scheduling under uncertainty).

Because the SRCPSP is known to be $\mathscr{NP}$-hard (Ballestìn, 2007), most researchers have focussed their efforts on developing heuristic solution methods. Golenko-Ginzburg and Gonik (1997) propose two heuristics that both rely on solving a series of multi-dimensional knapsack problems (the first heuristic uses an exact procedure whereas the second procedure resorts to a heuristic solution of the knapsack problems). Tsai and Gemmill (1998) apply simulated annealing and tabu search procedures, whereas Ballestín (2007) and Ballestín and Leus (2009) use a genetic algorithm and a GRASP algorithm respectively. Ashtiani et al. (2001) adopt a two-phase local-search procedure. Stork (2001), who builds on the work of Igelmund and Radermacher (1983) and Möhring (2000), is one of the few researchers that has developed exact procedures to optimally solve the SRCPSP. In his PhD, he compares the performance of five different branch-and-bound algorithms.

In this article, we extend the work of Creemers et al. (2010) and present an exact model that uses a backward stochastic dynamic-programming (SDP) recursion to determine the minimum expected-makespan of a resource-constrained project with stochastic activity durations. We use acyclic phase-type (PH) distributions to model activity durations and match the first two moments of the activity duration distributions. The complexity increases with the number of project activities and with decreasing levels of activity duration variability. Therefore, the model is intended for solving small-to medium-sized problem instances where activity durations have a moderate-to-high level of variability.

The main contributions of this article are: (1) we develop an exact and analytic solution method for optimally solving the SRCPSP, (2) our model drastically improves computational performance when compared to the model of Creemers et al. (2010), (3) our model outperforms the existing state-of-the-art when it comes to optimally solving small- to medium-sized problem instances where activities have a moderate-to-high level of duration variability, (4) our model significantly improves solution quality when compared to the heuristic approaches available in the literature, and (5) we investigate the impact of making scheduling decisions also during the execution of an activity rather than only at the end of an activity.

The remainder of this article is organized as follows. Sect. 2 presents the basic definitions and outlines the problem statement. The model itself is explained in Sect. 3. Sect. 4 provides a numerical example. The experiments and the results are discussed in Sect. 5. Sect. 6 concludes.

2 Definitions and problem statement

A project is a network of activities that can be represented by a graph $G = (V, E)$, where $V = \{0, 1, \ldots, n\}$ is a set of nodes and $E = \{(i, j)|i, j \in V\}$ is a set of arcs. The nodes represent project activities whereas the arcs represent precedence relationships. Activities 0 and n are dummy activities and represent the start and completion of the project respectively. The duration of an activity i is a random variable $\tilde{p}_i$ and has expected value μ_i and variance σ_i^2. $\tilde{\mathbf{p}} = \{\tilde{p}_0, \tilde{p}_1, \ldots, \tilde{p}_n\}$ denotes the vector of the activity duration random variables. p_i is a realization (or random variate) of $\tilde{p}_i$ and $\mathbf{p} = \{p_0, p_1, \ldots, p_n\}$ is a realization of $\tilde{\mathbf{p}}$. An activity i can start when all of its predecessors are finished and if sufficient resources are available. There are K renewable resource types. The availability of each resource type k is denoted by R_k. Each activity i requires $r_{i,k}$ units of resource k, where $r_{0,k} = r_{n,k} = 0$ for all $k \in \mathscr{R} = \{1, 2, \ldots, K\}$.

A solution to the basic type of RCPSP is a schedule $S = \{S_0, S_1, \ldots, S_n\}$, where S_i is the starting time of an activity i, $S_0 = 0$ (i.e., the project starts at time 0), and S_n represents the completion time (or makespan) of the project. In addition, define $\mathscr{A}(S, t) = \{i \in V : S_i \leq t \wedge (S_i + p_i) \geq t)$, the set of activities in schedule S that are active at time t. A schedule S is

feasible if:

$$S_i + p_i \leq S_j \qquad \forall (i,j) \in E, \tag{1}$$

$$\sum_{i \in \mathscr{A}(S,t)} r_{i,k} \leq R_k \qquad \forall t \geq 0, \forall k \in \mathscr{R}, \tag{2}$$

$$S_i \geq 0 \qquad \forall i \in V. \tag{3}$$

The optimal schedule S^* minimizes S_n subject to Constraints 1–3.

Because activity durations are not known in advance, a solution to the SRCPSP is a policy rather than a schedule. A policy Π is a set of decision rules that defines actions at decision times. Decision times are typically the start of the project and the completion times of activities. An action, on the other hand, corresponds to the start of a precedence- and resource-feasible set of activities. In addition, decisions have to respect the non-anticipativity constraint (i.e., a decision at time t can only use information that has become available before or at time t). As time progresses, activity duration realizations become known, and a schedule is generated (i.e., activities are assigned a starting time). Consequently, a policy Π may be interpreted as a function $\mathbb{R}_{\geq 0}^{n+1} \mapsto \mathbb{R}_{\geq 0}^{n+1}$ that maps given realizations of activity durations $\mathbf{p}$ to vectors of feasible starting times $S(\mathbf{p}; \Pi) = \{S_0(\mathbf{p}; \Pi), S_1(\mathbf{p}; \Pi), \ldots, S_n(\mathbf{p}; \Pi)\}$ (see for instance Igelmund and Radermacher (1983), Möhring (2000), and Stork (2001)). For a given realization $\mathbf{p}$ and policy Π, $S_n(\mathbf{p}; \Pi)$ denotes the makespan of schedule $S(\mathbf{p}; \Pi)$. The objective of the SRCPSP is to minimize $E(S_n(\mathbf{p}; \Pi))$ over a given class of policies (where $E(\cdot)$ is the expectation operator with respect to $\mathbf{p}$). Optimization over the class of all policies is computationally intractable. Therefore, most researchers focus their attention to the class of elementary policies $\mathcal{P}$, and allow decisions to be made only at the start of the project (i.e., at time 0) and at the completion times of activities.

3 Markov Decision Chain

A project network with stochastic activity durations is often referred to as a PERT network, and a PERT network with independent and exponentially-distributed activity durations is also called a Markovian PERT network. For Markovian PERT networks, Kulkarni and Adlakha (1986) have developed an exact method for deriving the distribution of the earliest project completion

time using a continuous-time Markov chain (CTMC). Buss and Rosenblatt (1997), Sobel et al. (2009), and Creemers et al. (2010) use the CTMC of Kulkarni and Adlakha (1986) as a starting point to develop scheduling procedures that maximize an expected-NPV (eNPV) objective. All aforementioned studies, however, assume unlimited resources and exponentially distributed activity durations. In this article, we extend the work of Creemers et al. (2010) to accommodate: (1) resource constraints, (2) PH-distributed activity durations, and (3) a minimum-makespan objective.

Below, we first study the special case of exponential activity durations (Sect. 3.1), followed by an overview of the PH distributions that are used in this article (Sect. 3.2). Next, we discuss how to incorporate PH distributions in the model (Sect. 3.3), and explain why they are a good choice for approximating the activity duration distributions (Sect. 3.4).

3.1 Exponential activity durations

In this section, we assume that duration $\tilde{p}_i$ is exponentially distributed with rate parameter $\lambda_i = \mu_i^{-1}$ for all $i \in V \setminus \{0, n\}$. At any time instant $t \geq 0$, the status of an activity is either idle (not yet started), ongoing (being executed), or finished (completed). Let $I(t)$, $O(t)$, and $F(t)$ denote the activities in V that are idle, ongoing, and finished at time t respectively. $I(t)$, $O(t)$, and $F(t)$ are mutually exclusive sets and $I(t) \cup O(t) \cup F(t) = V$ for all $t \geq 0$. Without loss of generality, we omit index t when referring to sets $I(t)$, $O(t)$, and $F(t)$.

The state of the system is defined by the status of the individual activities and can be represented by a pair (I, O) (set F can be obtained from sets V, I, and O). State transitions take place at the completion of an activity and are determined by the policy at hand. The starting conditions of the project are $I = V \setminus \{0\}$ and $O = \{0\}$. Selecting the optimal scheduling policy Π^* corresponds to minimizing a value function $G(\cdot)$ in a continuous-time Markov decision process (CTMDP) on the state space Q, with Q containing all feasible states. The value function $G(I, O)$ returns the expected time until completion of the project upon entry of state (I, O), conditional on the assumption that optimal decisions are made in all subsequent states (i.e., the Bellman principle of optimality applies).

Upon entry of state (I, O), a decision needs to be made whether or not to start a set of activities $W \subseteq H_{I,O}$, with $H_{I,O}$ the set of activities that are eligible to start. An activity i is eligible to start if:

1. $i \in I$,
2. $j \in F$ for all j for which $(j,i) \in E$,
3. $r_{i,k} \leq \left(R_k - \sum_{j \in O} r_{j,k} \right) \forall k \in \mathscr{R}$.

In addition, define $D(I,O,W)$, the time until completion of the project if a decision is made to start a set of activities W upon entry of state (I,O).

If no activities are started, a transition towards another state takes place after the first completion of an activity in O. The probability that an activity $i \in O$ finishes first equals $\lambda_i / \sum_{j \in O} \lambda_j$. The time until the first completion is exponentially distributed and has expected value $\left(\sum_{i \in O} \lambda_i \right)^{-1}$. Therefore, if no activities are started, the time until completion of the project upon entry of state (I,O) equals:

$$D(I,O,\emptyset) = \frac{1}{\sum\limits_{i \in O} \lambda_i} + \sum_{i \in O} \frac{\lambda_i}{\sum\limits_{j \in O} \lambda_j} G(I, O \setminus \{i\}). \tag{4}$$

If, on the other hand, a set of activities $W \subseteq H_{I,O}$ is started, an immediate transition towards another state is made and the time until completion of the project upon entry of state (I,O) equals:

$$D(I,O,W) = G(I \setminus W, O \cup W). \tag{5}$$

In order to determine the best set of activities to start, Creemers et al. (2010) enumerate all subsets of $H_{I,O}$, resulting in $2^{|H_{I,O}|}$ decisions to be evaluated. In this article, we propose a more efficient approach in which only $|H_{I,O}|$ decisions have to be evaluated. Each decision corresponds to the start of a single activity in $H_{I,O}$, and evaluating all $|H_{I,O}|$ decisions suffices to determine the optimal objective value upon entry of state (I,O). To see this, one only has to consider that: (1) upon starting an activity, an instantaneous transition is made towards a subsequent state and (2) due to the Bellman principle of optimality, optimal decisions are made in subsequent states. In other words, instead of starting multiple activities in a single instantaneous transition, we make multiple instantaneous transitions, during each of which a single activity is started. This modification results in a drastic reduction of the number of decisions to be evaluated upon entry of a state (I,O).

The impact of the modification on the computational performance of the backward SDP-recursion is verified in Sect. 5.1.

Upon entry of state (I, O), it is optimal to either not start activities or to start activity i^*:

$$i^* = \underset{i \in H_{I,O}}{\operatorname{argmin}} \left\{ D\left(I, O, \{i\}\right) \right\}. \tag{6}$$

Clearly, if $H_{I,O} = \emptyset$, the optimal decision is to not start activities and $G\left(I, O\right) = D\left(I, O, \emptyset\right)$. Otherwise, $G\left(I, O\right)$ equals:

$$G\left(I, O\right) = \min \left\{ D\left(I, O, \emptyset\right), D\left(I, O, \{i^*\}\right) \right\}. \tag{7}$$

The backward SDP-recursion starts in $(I, O) = (\{n\}, \emptyset)$ and stops if $(I, O) = (V \setminus \{0\}, \{0\})$, and the optimal objective value equals $\min E\left(S_n\left(\mathbf{p}; \Pi^*\right)\right) = G(V \setminus \{0\}, \{0\})$.

3.2 PH distributions

In this article, we model activity durations using acyclic, continuous-time PH distributions. Continuous-time PH distributions use exponentially-distributed building blocks to approximate any positive-valued continuous distribution with arbitrary precision (see Neuts (1981), Latouche and Ramaswami (1999), and Osogami (2005) for further details on PH distributions). More formally, a PH distribution is the distribution of time until absorption in a Markov chain with absorbing state 0 and state space $\{0, 1, \ldots, Z\}$. It is fully characterized by parameters $\boldsymbol{\tau}$ and $\mathbf{Z}$. $\boldsymbol{\tau}$ is the vector of probabilities to start the process in any of the Z transient states (i.e., phases) and $\mathbf{Z}$ is the transient state transition matrix. The infinitesimal generator of the Markov chain representing the PH distribution is:

$$\mathbf{Q} = \begin{pmatrix} 0 & \mathbf{0} \\ \mathbf{t} & \mathbf{Z} \end{pmatrix},$$

where $\mathbf{0}$ is a zero matrix and $\mathbf{t} = -\mathbf{Z}\mathbf{e}$ (with $\mathbf{e}$ a vector of ones).

Various techniques exist to approximate a given distribution by means of a PH distribution. In this article, we adopt a two-moment matching procedure that minimizes the required number of phases. Let ν_i denote the squared coefficient of variation (SCV) of the duration of activity i:

$$\nu_i = \sigma_i^2 \mu_i^{-2}. \tag{8}$$

We define three cases: (1) $\nu_i = 1$, (2) $\nu_i > 1$, and (3) $\nu_i < 1$. In the first case, we approximate the activity duration distribution by means of an exponential distribution with rate parameter $\lambda_i = \mu_i^{-1}$. The parameters of the corresponding PH distribution are:

$$\begin{aligned} \boldsymbol{\tau}_i &= 1, \\ \mathbf{Z}_i &= (-\lambda_i). \end{aligned}$$

In the second case, we use a two-phase Coxian distribution where the rate parameter of the first phase is determined by means of a scaling factor κ:

$$\lambda_{i,1} = \frac{1}{\mu_i \kappa}. \tag{9}$$

The expected value of the two-phase Coxian distribution is:

$$\mu_i = \lambda_{i,1}^{-1} + \pi_{i,1,2}\lambda_{i,2}^{-1}, \tag{10}$$

where $\lambda_{i,2}$ is the exponential rate parameter of the second phase and $\pi_{i,1,2}$ is the probability of visiting the second phase. The variance of the two-phase Coxian distribution is:

$$\sigma_i^2 = \lambda_{i,1}^{-2} + 2\pi_{i,1,2}\lambda_{i,2}^{-2} - \pi_{i,1,2}^2\lambda_{i,2}^{-2}. \tag{11}$$

When deriving parameters $\lambda_{i,2}$ and $\pi_{i,1,2}$ as a function of parameters μ_i, ν_i and κ, we obtain:

$$\lambda_{i,2} = \frac{2(\kappa - 1)}{\mu_i(2\kappa - 1 - \nu_i)}, \tag{12}$$

$$\pi_{i,1,2} = \frac{2(\kappa - 1)^2}{1 + \nu_i - 2\kappa}. \tag{13}$$

The parameters of the corresponding PH distribution are:

$$\begin{aligned} \boldsymbol{\tau}_i &= \mathbf{e}_1, \\ \mathbf{Z}_i &= \begin{pmatrix} -\lambda_{i,1} & \pi_{i,1,2}\lambda_{i,1} \\ 0 & -\lambda_{i,2} \end{pmatrix}, \end{aligned}$$

where $\mathbf{e}_1$ is a single-entry vector that is populated with zeroes except for the first entry, which equals one. In the third case, we use a hypo-exponential

distribution (a series of exponential distributions whose parameters are allowed to differ; a generalization of the Erlang distribution). The number of required phases equals:

$$Z_i = \lceil \nu_i^{-1} \rceil. \tag{14}$$

We assume that the first $Z_i - 1$ phases of the hypo-exponential distribution are independent and identically exponentially distributed with rate parameter $\lambda_{i,1} = \lambda_{i,2} = \ldots = \lambda_{i,Z_i-1}$. The last phase is exponentially distributed with rate parameter λ_{i,Z_i}. The expected value and variance of the hypo-exponential distribution are:

$$\mu_i = \sum_{z=1}^{Z_i} \lambda_{i,z}^{-1}, \tag{15}$$

$$\sigma_i^2 = \sum_{z=1}^{Z_i} \lambda_{i,z}^{-2}. \tag{16}$$

When deriving the rate parameters of the hypo-exponential distribution as a function of parameters μ_i, ν_i and Z_i, we obtain:

$$\lambda_{i,Z_i} = \frac{1 + \sqrt{(Z_i - 1)\left(Z_i \nu_i - 1\right)}}{\mu_i \left(1 - Z_i \nu_i + \nu_i\right)}, \tag{17}$$

$$\lambda_{i,z} = \frac{(Z_i - 1) - \sqrt{(Z_i - 1)\left(Z_i \nu_i - 1\right)}}{\mu_i \left(1 - \nu_i\right)}, \tag{18}$$

for all $z \in \{1, 2, \ldots, Z_i - 1\}$. The parameters of the corresponding PH distribution are:

$$\boldsymbol{\tau}_i = \mathbf{e}_1,$$

$$\mathbf{Z}_i = \begin{pmatrix} -\lambda_{i,1} & \lambda_{i,1} & 0 & \cdots & 0 & 0 \\ 0 & -\lambda_{i,2} & \lambda_{i,2} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\lambda_{i,Z_i-1} & \lambda_{i,Z_i-1} \\ 0 & 0 & 0 & \cdots & 0 & -\lambda_{i,Z_i} \end{pmatrix}.$$

For the three cases, Z_i equals 1, 2, and $\lceil \nu_i^{-1} \rceil$ respectively. Figure 1 provides an overview of the PH distributions that are used in this article. Note, however, that our method works with all acyclic PH distributions.

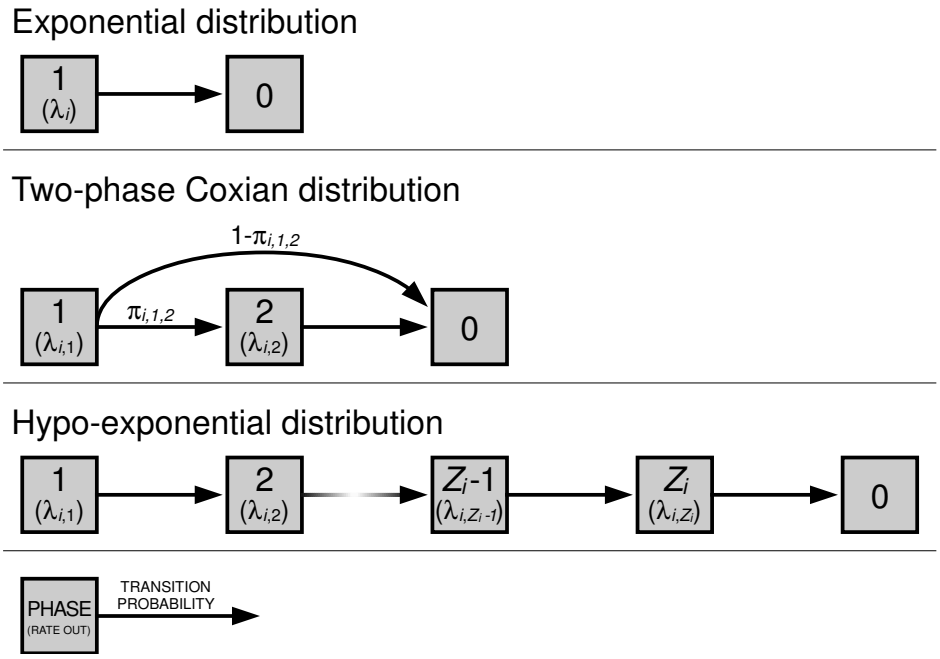

Figure 1: Overview of PH distributions

3.3 PH-distributed activity durations

In this section, we describe how to obtain the minimum expected-makespan of a resource-constrained project when activity durations are PH distributed. The duration of an activity $i \in V \setminus \{0, n\}$ can be seen as a sequence of Z_i phases where each phase $\theta_{i,z}$: (1) has an exponential duration with rate parameter $\lambda_{i,z}$, (2) has a probability $\tau_{i,z}$ to be the initial phase when starting activity i, and (3) is visited with probability $\pi_{i,y,z}$ when departing from another phase $\theta_{i,y}$. Also note that, due to the acyclic nature of the adopted PH distributions, a phase can never be visited more than once.

The use of PH-distributed activity durations makes it possible to transform any project network into a Markovian PERT network. Therefore, given a few adaptations, the SDP recursion described in Sect. 3.1 can easily be extended to accommodate PH distributions. The most important adaptation concerns the status of the ongoing activities. If activity durations are PH distributed, we not only have to keep track of the ongoing activities themselves, but we also need to keep track of their ongoing phase. Let $(I, \mathcal{O})$ denote the state of the system, where $\mathcal{O}$ is the set of ongoing phases. Upon completion of a phase $\theta_{i,y} \in \mathcal{O}$:

1. Activity i completes with probability $\pi_{i,y,0}$ (i.e., the probability of being absorbed when departing from phase $\theta_{i,y}$), and a transition is made towards state $(I, \mathcal{O} \setminus \{\theta_{i,y}\})$.

2. Activity i does not complete, phase $\theta_{i,z}$ is started with probability $\pi_{i,y,z}$, and a transition is made towards state $(I, \mathcal{O} \cup \{\theta_{i,z}\} \setminus \{\theta_{i,y}\})$.

Note that $F = V \setminus (I \cup O)$, and O can be obtained from $\mathcal{O}$ as follows:

$$O = \{i | \mathcal{Z}_i \cup \mathcal{O} \neq \emptyset\}, \tag{19}$$

where $\mathcal{Z}_i = \{\theta_{i,1}, \theta_{i,2}, \ldots, \theta_{i,Z_i}\}$.

let $H_{I,\mathcal{O}}$ denote the set of activities that are eligible to start upon entry of state $(I, \mathcal{O})$ (its definition is analogous to that of $H_{I,O}$). If no activities are started, the time until completion of the project upon entry of state $(I, \mathcal{O})$ equals:

$$\begin{aligned} D(I, \mathcal{O}, \emptyset) = \frac{1}{\lambda_{\mathcal{O}}} + \sum_{\theta_{i,y} \in \mathcal{O}} \frac{\lambda_{i,y}}{\lambda_{\mathcal{O}}} \pi_{i,y,0} G(I, \mathcal{O} \setminus \{\theta_{i,y}\}) + \\ \sum_{\theta_{i,y} \in \mathcal{O}} \frac{\lambda_{i,y}}{\lambda_{\mathcal{O}}} \sum_{z=y+1}^{Z_i} \pi_{i,y,z} G(I, \mathcal{O} \cup \{\theta_{i,z}\} \setminus \{\theta_{i,y}\}), \end{aligned} \tag{20}$$

where $\lambda_{\mathcal{O}} = \sum_{\theta_{j,z} \in \mathcal{O}} \lambda_{j,z}$.

If $H_{I,\mathcal{O}} \neq \emptyset$, the best activity to start is:

$$i^{\star} = \underset{i \in H_{I,\mathcal{O}}}{\operatorname{argmin}} \left\{ \sum_{z=1}^{Z_i} \tau_{i,z} G\left(I \setminus \{i\}, \mathcal{O} \cup \{\theta_{i,z}\}\right) \right\}. \tag{21}$$

The optimal objective value upon entry of state $(I, \mathcal{O})$ equals:

$$G\left(I, \mathcal{O}\right) = \min\left\{D\left(I, \mathcal{O}, \emptyset\right), D\left(I, \mathcal{O}, \{i^{\star}\}\right)\right\}. \tag{22}$$

Note that, after the completion of a phase, it is possible to interrupt the execution of an activity and/or to make scheduling decisions. Therefore, using PH-distributed activity durations, it becomes possible to optimize over a class of policies that is far more general than $\mathcal{P}$. Let $\mathcal{N}$ denote the class of policies where decisions can be made: (1) at the start of the project, (2) at the completion times of activities, and (3) at the completion times of activity phases. It is clear that $\mathcal{N}$ dominates $\mathcal{P}$. In Sect. 5.5, we evaluate the gain in solution quality when we optimize over $\mathcal{N}$ rather than over $\mathcal{P}$.

3.4 Why PH distributions?

In most cases, the "true" distribution of the duration of an activity is unknown. Often, it is assumed that the duration of an activity follows a beta, uniform, or Gaussian distribution (see for instance Herroelen and Leus (2005), Bidot et al. (2009), Fu et al. (2012), and Creemers et al. (2014)). These distributions, however, only allow to match the first two moments of the true duration distribution. PH distributions, on the other hand, can match almost any distribution with arbitrary precision. Therefore, one could argue that PH distributions are a better choice as an approximation of the true duration distribution.

The size of a Markovian PERT network is determined by the number of phases that is required to model the project activities. Therefore, PH distributions are especially suited to approximate activity durations that have a moderate-to-high level of variability. In fact, any activity duration distribution that has a SCV in $[0.5, \infty)$ can be modeled using a PH distribution of up to two phases. For SCV smaller than 0.5, the number of required phases increases rapidly, making PH distributions less applicable for settings where activity duration variability is small. It mainly makes sense, however, to

solve the SRCPSP if activity durations exhibit a sufficient degree of variability. If activity durations have a low level of variability, solving the SRCPSP has limited added value, and the RCPSP may serve as an approximation of the SRCPSP.

As is explained in Sect. 3.2, PH distributions are a mixture of exponential distributions. Due to the memoryless property of the exponential distribution, there is no need to keep track of the remaining duration of the ongoing activities (i.e., the remaining duration of a phase $\theta_{i,z}$ is exponentially distributed with rate parameter $\lambda_{i,z}$ for every moment in time at which phase $\theta_{i,z}$ is ongoing). As a result, the state of the system is fully defined by the set of idle activities and the set of ongoing activity phases. This compact representation of the state space allows us to solve problem instances which are beyond reach for other optimal solution methods.

If activity durations are PH distributed, it becomes possible to interrupt the execution of an activity and/or to make scheduling decisions at the completion of a phase rather than only at the completion of an activity. This enables the study of more complex scheduling problems and also allows to optimize over a class of policies that is more general than the class of elementary policies $\mathcal{P}$.

4 Example

Any project network with stochastic activity durations can be transformed into a Markovian PERT network. In order to illustrate this transformation process, we consider a project that consists of five activities (i.e., $V = \{0, 1, 2, 3, 4\}$, where 0 and 4 are dummy activities). The data of the project is summarized in Table 1. The project network and its transformation are presented in Figure 2. Activity 1 has duration SCV equal to $1/3$ and can be modeled as a series of three phases that have exponentially distributed durations (i.e., a hypo-exponential distribution is used). Activity 2 has duration SCV equal to 1 and can be modeled as a single phase (i.e., the duration of activity 2 is approximated by an exponential distribution). If the activity duration SCV is larger than 1, a two-phase Coxian distribution is used. Activity 3, for example, has a duration SCV equal to 2 and can be modeled as a series of two phases, where the second phase is executed with probability $\pi_{3,1,2}$. Note that this implies that not all phases have to be executed in order to complete the project (i.e., there is a probability $\pi_{3,1,0}$ of

Table 1: Data for the example project

i	p_i	ν_i	$r_{i,1}$	Z_i	$\lambda_{i,1}$	$\lambda_{i,2}$	$\pi_{i,1,2}$
0	0	0	0				
1	9	1/3	5	3	1/3	1/3	
2	9	1	5	1	1/9		
3	10	2	5	2	1/5	1/20	1/4
4	0	0	0				
R_1	10						

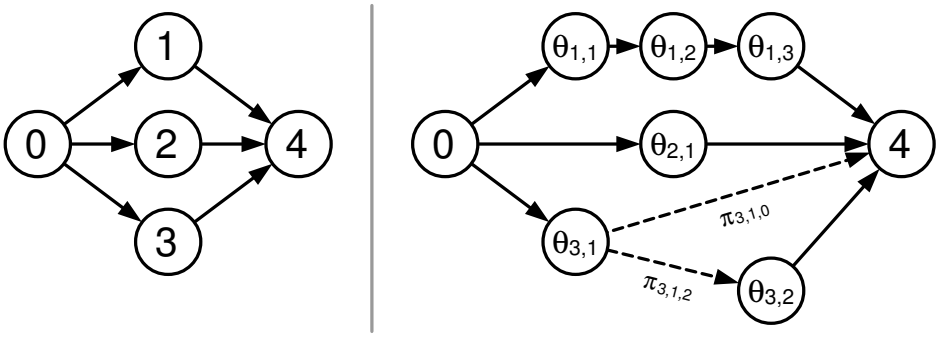

Figure 2: Example project network and its corresponding Markovian PERT network

not having to execute the second phase of activity 3).

Next, we use the example project to illustrate the importance of stochastic activity durations when solving the RCPSP. In the example, we assume that there is a single resource (i.e., $K = 1$) that has an availability of 10 resource units (i.e., $R_1 = 10$). The non-dummy activities each consume 5 resources (i.e., $r_{1,1} = r_{2,1} = r_{3,1} = 5$). If activity durations are deterministic, the optimal policy is to start activities 2 and 3, and to start activity 1 after completion of either activity 2 or 3. Refer to this policy as Π_1. Adopting policy Π_1 results in a project makespan of 18 time units if activity durations are deterministic. Policy Π_2, on the other hand, starts activities 1 and 2, and starts activity 3 after completion of either activity 1 or 2. If activity durations are deterministic, policy Π_2 corresponds to a makespan of 19 time units. Figure 3 illustrates both policies. If activity durations are not deterministic, however, policy Π_2 may outperform policy Π_1. This is illustrated in Figure 4,

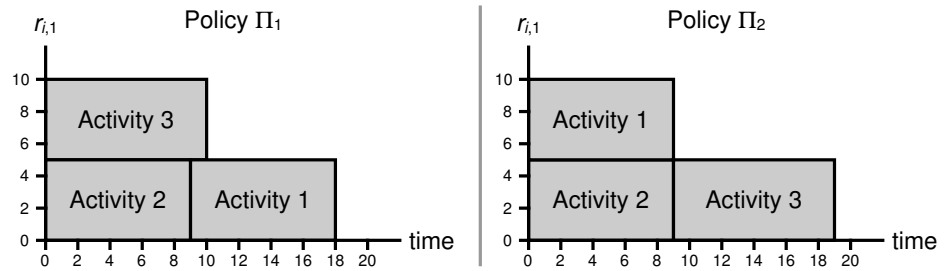

Figure 3: Illustration of policy Π_1 and Π_2 if activity durations are deterministic

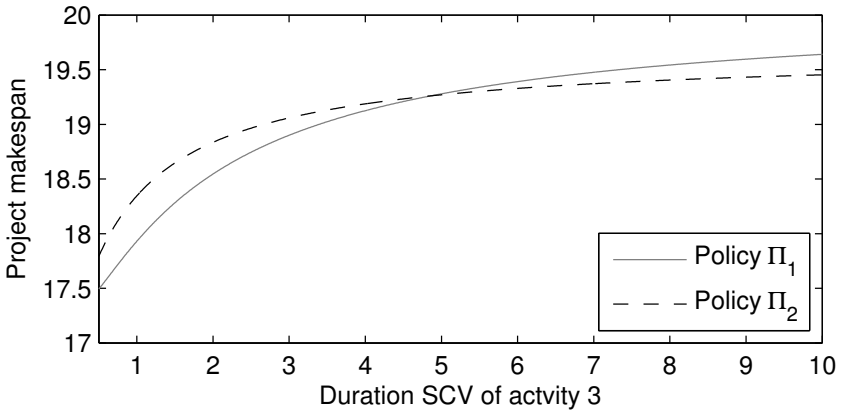

Figure 4: Project makespan of policy Π_1 and Π_2 as a function of the SCV of the duration of activity 3

which shows the makespan of the project as a function of the variability of the duration of activity 3. It is clear that, as the variability of the duration of activity 3 increases, policy Π_2 becomes more effective when compared to policy Π_1. For a duration SCV larger than 4.87 (i.e., for $\nu_3 > 4.87$), the makespan that corresponds to policy Π_2 is smaller than the makespan that is associated with policy Π_1.

5 Results

In this section, we assess the difference in performance between our approach and the approach of Creemers et al. (Sect. 5.1). Next, we discuss the different

problem sets that are used in the literature (Sect. 5.2), and we compare the computational performance of our model and other optimal approaches (Sect. 5.3). We also evaluate the optimality gap of the existing heuristic procedures (Sect. 5.4), and investigate the impact of making decisions also during the execution of an activity (Sect. 5.5).

All our experiments are performed on an AMD Phenom II 3.2 GHz computer with 32,768 MB RAM.

5.1 Improving the model of Creemers et al. (2010)

Creemers et al. (2010) use a backward SDP-recursion to find the maximum eNPV of a project where activity durations are exponentially distributed. In this article, we modify this SDP recursion to accommodate: (1) resource constraints, (2) PH-distributed activity durations, and (3) a minimum-makespan objective. Next to structural changes, however, we also suggest a modification that drastically reduces the number of decisions that have to be evaluated upon entry of a state (see Sect. 3 for more details). In this section, we assess the impact of this modification on the computational performance of the backward SDP-recursion. For this purpose, we replicate the study of Creemers et al. (2010).

In their study, Creemers et al. (2010) generate 30 scheduling instances for each combination of order strength (OS) and network size. The OS is either 0.4, 0.6, or 0.8. The size of the network ranges from 10 to 120 activities. In total, $1,080$ instances are generated. We analyze these instances using (1) the SDP recursion of Creemers et al. (2010) and (2) the modified SDP recursion that is introduced in Sect. 3.3. The results are presented in Table 2. For each combination of parameters, Table 2 aggregates: (1) the number of instances that were solved to optimality, (2) the CPU time of the "old" approach, and (3) the CPU time of the "new" approach. It is clear that the modified approach drastically improves computation times. In fact, on average, the computational efficiency has been improved by a factor of 56.49.

5.2 Data sets used in the literature on the SRCPSP

Various data sets are available in the literature. Tsai and Gemmill (1998), Ballestín and Leus (2009), and Ashtiani et al. (2011) assess the performance of their procedures using the instances of the Patterson data set (Patterson 1984). Stork (2001) evaluates his branch-and-bound algorithms on the J30

Table 2: Comparison of the computational results when using the SDP recursion of Creemers et al. (2010) and the modified SDP recursion

	Solved successfully			Average CPU time (old)			Average CPU time (new)		
n	OS = 0.8	OS = 0.6	OS = 0.4	OS = 0.8	OS = 0.6	OS = 0.4	OS = 0.8	OS = 0.6	OS = 0.4
10	30	30	30	0.00	0.00	0.00	0.00	0.00	0.00
20	30	30	30	0.00	0.01	0.46	0.00	0.00	0.03
30	30	30	30	0.01	0.33	26.92	0.00	0.02	1.49
40	30	30	29	0.03	6.62	2,337.96	0.00	0.49	72.25
50	30	30	4	0.15	100.28	52,267.30	0.01	4.43	823.71
60	30	30	0	0.74	2,210.08	-	0.06	67.87	-
70	30	22	0	3.19	17,495.49	-	0.24	378.64	-
80	30	9	0	10.81	72,473.41	-	0.79	1,188.01	-
90	30	0	0	50.64	-	-	3.15	-	-
100	30	0	0	171.42	-	-	9.60	-	-
110	30	0	0	1,193.88	-	-	40.93	-	-
120	30	0	0	12,789.06	-	-	260.66	-	-

and J60 instances of the well-known PSPLIB data set (Kolisch and Sprecher 1996). Ballestín and Leus (2009) and Ashtiani et al. (2011) use the J120 instances of the same data set. Golenko-Ginzburg and Gonik (1997) use a single instance with 36 activities to evaluate their two heuristics. The same problem instance is also used in Ballestín and Leus (2009) and Ashtiani et al. (2011). In our experiments, we use the problem instances of the Patterson data set and the J30 and J60 instances of the PSPLIB data set. We do not use the J120 instances of the PSPLIB data set because they cannot be solved to optimality. We also do not use the example project presented in Golenko-Ginzburg and Gonik (1997) because its activities have a very limited duration variability (e.g., when the uniform distribution is used to model activity durations, the average duration SCV equals 0.014).

5.3 Computational performance and comparison with optimal procedures

In this section, we discuss the computational performance of our model and compare with other optimal approaches available in the literature. In a first experiment, we assume that activity durations are exponentially distributed and solve the instances of the Patterson data set and the J30 and J60 instances of the PSPLIB data set. Table 3 summarizes the results (the state-space sizes are expressed in thousands of states) and Figure 5 presents a box plot of the computation times. It is clear that project networks of up to 32 activities are analyzed with ease. The results also show that networks of 62 activities can often be solved (we solve 301 out of 480 networks), albeit at a larger computational cost.

Next, we use the J30 instances of the PSPLIB data set to analyze the impact of different levels of activity duration variability on the performance of our model. Table 4 summarizes the results (the state-space sizes are expressed in thousands of states). The level of activity duration variability determines the number of required phases. For values of SCV larger than 0.5, one or two phases suffice. If, however, the SCV of the activity durations is smaller than 0.5, the number of required phases increases rapidly. As a result, the size of the state space increases exponentially and computational performance plummets. For moderate-to-high levels of activity duration variability, however, the computational effort is acceptable.

The main bottleneck of our approach is memory rather than CPU time.

Table 3: Computational results if activity durations are exponentially distributed (state-space sizes are expressed in thousands of states)

Data set	Patterson	J30	J60
Instances in set	110	480	480
Instances solved	110	480	301
Avg # of activities	26	32	62
Avg CPU time (s)	0.00	0.49	1,564
Max CPU time (s)	0.05	13.1	31,838
Min CPU time (s)	0.00	0.00	1.90
Avg state-space size	7.45	539	661,315
Max state-space size	136	11,378	4,257,393
Min state-space size	0.03	6.17	3,762

Table 4: Computational results for different values of SCV when solving the J30 instances of the PSPLIB data set (state-space sizes are expressed in thousands of states)

Average SCV	Instances solved	CPU time (s)			State-space size		
		min	avg	max	min	avg	max
1/4	358	0.08	28.32	217.20	181	42,702	580,059
1/3	421	0.03	24.00	593.29	126	66,134	1,092,331
1/2	480	0.02	28.54	1453.03	79	89,863	3,000,505
1	480	0.00	0.49	14.02	6	539	11,378
2	480	0.03	34.83	1731.67	79	89,863	3,000,505

For large networks and/or low levels of activity duration variability, the state space becomes too big to store in memory. As a result, our model is mainly suited for small-to medium-sized projects where activity durations have a moderate-to-high level of variability. For this setting, our approach outperforms the current state-of-the-art.

The literature on optimal solution methods for the SRCPSP is rather scarce. With respect to the Patterson data set, Tsai and Gemmill (1998) are able to solve 95 out of 110 instances to optimality if activity durations are deterministic. If activity durations are stochastic, optimality cannot be guaranteed. With respect to the J30 and J60 instances of the PSPLIB data set, Stork (2001) is able to optimally solve 179 and 11 out of 480 instances respectively. It is clear that our model performs better.

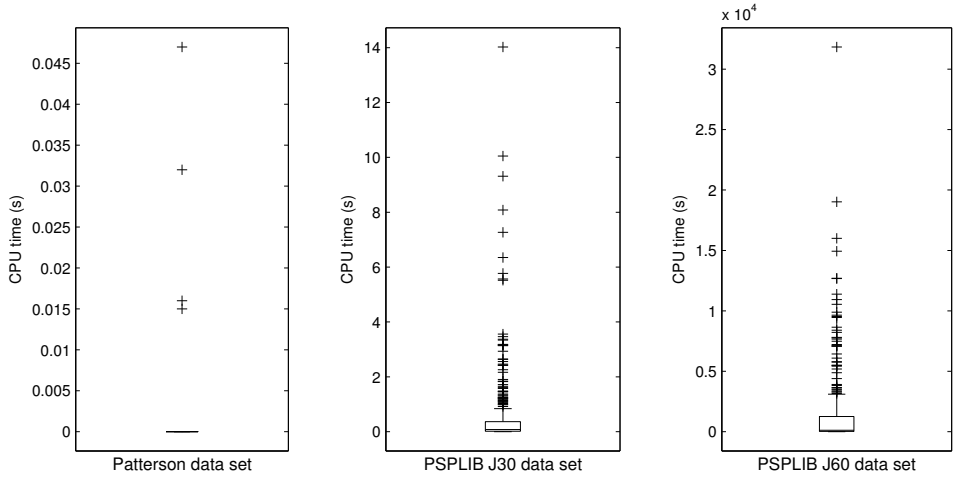

Figure 5: Computational performance if activity durations are exponentially distributed

5.4 Comparison with heuristic procedures

In this section, we assess the optimality gap of the heuristic approaches that are available in the literature. The models of Ballestìn and Leus (2009) and Ashtiani et al. (2001) are the current state-of-the-art when it comes to heuristically solving the SRCPSP. Both authors report results on the J120 data set. Currently, however, it is impossible to optimally solve the instances of that data set (i.e., the optimality gap cannot be evaluated). If activity durations are exponentially distributed, our model can optimally solve 480 and 301 instances of the J30 and J60 data sets respectively (see Sect. 5.3 for more details). For these instances, we can measure the optimality gap if we also have the solutions of the heuristic approaches. Unfortunately, the solutions are not available from Ashtiani et al. (2001). We are able, however, to compare with the solutions of Ballestìn and Leus (2009).

We assess the optimality gap of the GRASP method of Ballestìn and Leus (2009) with a limit of 25,000 schedules. The results are reported in Table 5. Table 5 presents the minimum, average, and maximum difference between the optimal makespan and the the makespan produced by the GRASP procedure. In addition, Figure 6 presents a boxplot of the optimality gap. We

Table 5: Optimality gap of the GRASP method if 25,000 schedules are used and activity durations are exponentially distributed

	Instances	Optimality gap		
Data set	solved	min	avg	max
J30	480	1.07%	9.11%	20.20%
J60	301	9.61%	15.88%	24.80%

find that our model improves solution quality with 9.11% on average for the J30 instances, and with 15.88% on average for the J60 instances. It is clear that the optimality gap increases with increasing network size. Therefore, we expect that the gap for the J120 instances is even larger. In addition, the minimum optimality gap increases rather drastically, indicating that it becomes more difficult for the heuristic approaches to approximate the optimal solution as the size of the network increases.

5.5 Value of non-elementary policies

All solution methods in the literature on the SRCPSP allow decisions to be taken only at the start of the project and at the completion times of activities (i.e., they optimize over the class of elementary policies $\mathcal{P}$). In this section, we investigate the difference in solution quality if we allow decisions to be taken also at the completion of an activity phase (i.e., we observe the impact of optimizing over the class of non-elementary policies $\mathcal{N}$).

For this experiment, we use the 110 instances of the Patterson data set. The average SCV of the activity durations ranges from 0.1 to 2.0. Depending on the SCV, the number of phases differs, and hence, the number of decision moments during the execution of an activity differs as well. Table 5.5 presents the difference in solution quality when optimizing over $\mathcal{N}$ rather than over $\mathcal{P}$. It is clear that the difference is minimal (at most, the difference amounts to 0.55 %). This, however, is not really surprising as it is often optimal to start activities as soon as possible if makespan is to be minimized (i.e., there is limited value in postponing the start of an activity). We do observe, however, that the difference in solution quality grows larger if activity duration variability increases (for a constant number of phases/decision moments). This indicates that it is more beneficial to have decision moments during the execution of an activity if the duration of that activity is more

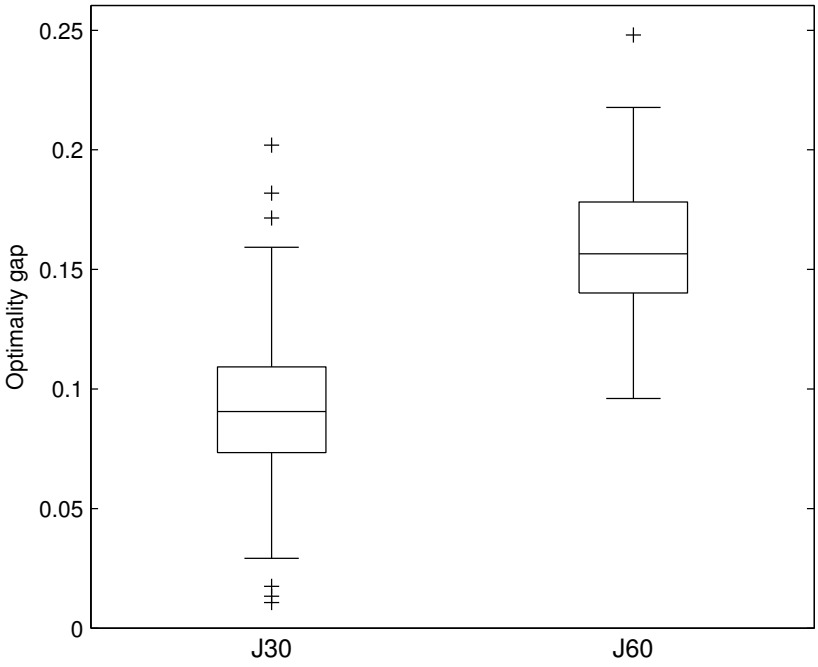

Figure 6: Optimality gap of the GRASP method if 25,000 simulations are used and activity durations are exponentially distributed

Table 6: Percentual difference in solution quality for policy classes $\mathcal{P}$ and $\mathcal{N}$ for different values of SCV

Average SCV	Number of phases	Average difference	Maximum difference
0.1	10	0.0016%	0.015%
0.2	5	0.0015%	0.017%
0.3	4	0.0020%	0.015%
0.4	3	0.0018%	0.031%
0.5	2	0.0006%	0.006%
0.6	2	0.0014%	0.013%
0.7	2	0.0016%	0.018%
0.8	2	0.0014%	0.016%
0.9	2	0.0010%	0.010%
1.0	1	0.0000%	0.000%
1.1	2	0.0009%	0.029%
1.2	2	0.0024%	0.066%
1.3	2	0.0042%	0.103%
1.4	2	0.0071%	0.115%
1.5	2	0.0102%	0.215%
1.6	2	0.0130%	0.319%
1.7	2	0.0154%	0.399%
1.8	2	0.0174%	0.462%
1.9	2	0.0191%	0.511%
2.0	2	0.0205%	0.550%

variable.

6 Conclusions

In this article, we have presented an exact and analytic solution method for optimally solving the SRCPSP. Our model extends the SDP recursion of Creemers et al. (2010) and accommodates: (1) resource constraints, (2) PH-distributed activity durations, and (3) a minimum-makespan objective. Next to these structural improvements, we also improve the computational efficiency of the SDP recursion by a factor of 56.49 on average.

Experiments have shown that our model performs best when activity durations have a moderate-to-high level of variability, and that it can be used to optimally solve project instances that have up to 62 activities. For this setting, our model outperforms the existing state-of-the-art.

We have also used our model to assess the optimality gap of the heuristic approaches available in the literature. We show that our model improves the solution quality of the GRASP procedure of Ballestìn and Leus (2009) with 9.11% and 15.88% on average for instances that have 32 and 62 activities respectively. This indicates that it becomes more difficult for the heuristic approaches to approximate the optimal solution as the size of the network increases.

In addition, we have investigated the difference in solution quality if we allow scheduling decisions to be made at the end of an activity phase rather than only at the end of an activity. An experiment has shown that the difference in solution quality is minimal (i.e., there is limited value in postponing the start of an activity). The experiment also shows that it is more beneficial to have decision moments during the execution of an activity if the duration of that activity is more variable.

Last, we have also illustrated that variability in activity durations is an important factor when solving the RCPSP. As such, it might not be advisable to assume that activity durations are deterministic when making project scheduling decisions.

References

[1] Ashtiani, B., Leus R., & Aryanezhad, M.B. (2011). New competitive results for the stochastic resource-constrained project scheduling problem: Exploring the benefits of pre-processing. *Journal of Scheduling*, *14*(2), 157-171.

[2] Ballestín, F. (2007). When it is worthwhile to work with the stochastic RCPSP? *Journal of Scheduling*, *10*(3), 153–166.

[3] Ballestín, F., & Leus, R. (2009). Resource-constrained project scheduling for timely project completion with stochastic activity durations. *Production and Operations Management*, *18*(4), 459–474.

[4] Bidot, J., Vidal, T., Laborie, P., & Beck, J.C. (2009). A theoretic and practical framework for scheduling in a stochastic environment. *Journal of Scheduling*, *12*(3), 315–344.

[5] Buss, A.H., & Rosenblatt, M.J. (1997). Activity delay in stochastic project networks. *Operations Research*, *45*(1), 126–139.

[6] Creemers, S., Leus, R., & Lambrecht, M. (2010). Scheduling Markovian PERT networks to maximize the net present value. *Operations Research Letters*, *38*(1), 51–56.

[7] Creemers, S., Demeulemeester, E., & Van de Vonder, S. (2014). A new approach for quantitative risk analysis. *Annals of Operations Research*, *213*(1), 27–65.

[8] Debels, D., & Vanhoucke, M. (2007). A decomposition-based genetic algorithm for the resource-constrained project-scheduling problem. *Operations Research*, *55*(3), 457-469.

[9] Demeulemeester, E., & Herroelen, W. (2002). *Project Scheduling: A Research Handbook*. AH Dordrecht: Kluwer Academic Publishers Group.

[10] Fernandez, A.A., Armacost, R.L., & Pet-Edwards, J. (1996). The role of the non-anticipativity constraint in commercial software for stochastic project scheduling. *Computers and Industrial Engineering*, *31*(1), 233-236.

[11] Fu, N., Lau, H.C., Varakantham, P., & Xiao, F. (2012). Robust local search for solving RCPSP/max with durational uncertainty *Journal of Artificial Intelligence Research*, *43*, 43-86.

[12] Golenko-Ginzburg, D., & Gonik, A. (1997). Stochastic network project scheduling with non-consumable limited resources. *International Journal of Production Economics*, *48*(1), 29–37.

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

[14] Igelmund, G., & Radermacher, F.J. (1983). Preselective strategies for the optimization of stochastic project networks under resource constraints. *Networks*, *13*(1), 1–28.

[15] Kolisch, R., & Sprecher, A. (1996). PSPLIB - A project scheduling problem library. *European Journal of Operational Research*, *96*(1), 205-216.

[16] Kulkarni, V., & Adlakha, V. (1986). Markov and Markov-regenerative PERT networks. *Operations Research*, *34*(5), 769–781.

[17] Latouche, G., & Ramaswami, V. (1999). *Introduction to Matrix Analytic Methods in Stochastic Modeling*. Philadelphia: American Statistical Association and the Society for Industrial and Applied Mathematics.

[18] Möhring, R.H. (2000). Scheduling under uncertainty: Optimizing against a randomizing adversary. *Lecture Notes in Computer Science*, *1913*, 15–26.

[19] Neumann, K., Schwindt, C., & Zimmermann, J. (2003). *Project Scheduling with Time Windows and Scarce Resources*. Berlin: Springer-Verlag.

[20] Neuts, M.F. (1981). *Matrix-geometric solutions in stochastic models*. Baltimore: Johns Hopkins University Press.

[21] Osogami, T. (2005). *Analysis of multiserver systems via dimensionality reduction of Markov chains*. PhD thesis, Carnegie Mellon University.

[22] Patterson, J.H. (1984). A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem. *Management Science*, *30*(7), 854–867.

[23] Sobel, M.J., Szmerekovsky, J.G., & Tilson, V. (2009). Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, *198*(1), 697–705.

[24] Stork, F. (2001). *Stochastic Resource-Constrained Project Scheduling*. PhD thesis, Technische Universität Berlin.

[25] Tsai, Y.-W., & Gemmill, D.D. (1998). Using tabu search to schedule activities of stochastic resource-constrained projects. *European Journal of Operational Research*, *111*(1), 129–141.