doi:10.1016/j.ejor.2017.11.027 • www.stefancreemers.be • info@stefancreemers.be

Maximizing the expected net present value of a project with phase-type distributed activity durations: an efficient globally optimal solution procedure

Stefan Creemers

Abstract - **We study projects with activities that have stochastic durations that are modeled using phase-type distributions. Intermediate cash flows are incurred during the execution of the project. Upon completion of all project activities a payoff is obtained. Because activity durations are stochastic, activity starting times cannot be defined at the start of the project. Instead, we have to rely on a policy to schedule activities during the execution of the project. The optimal policy schedules activities such that the expected net present value of the project is maximized. We determine the optimal policy using a new continuous-time Markov chain and a backward stochastic dynamic program. Although the new continuous-time Markov chain allows to drastically reduce memory requirements (when compared to existing methods), it also allows activities to be preempted; an assumption that is not always desirable. We prove, however, that it is globally optimal not to preempt activities if cash flows are incurred at the start of an activity. Moreover, this proof holds regardless of the duration distribution of the activities. A computational experiment shows that we significantly outperform current state-of-the-art procedures. On average, we improve computational efficiency by a factor of 600, and reduce memory requirements by a factor of 321.**

Keywords - **project scheduling, project management, NPV maximization, SNPV, stochastic activity durations**

1 Introduction

Most of the literature on project scheduling has focussed on minimizing the makespan of a project, that is, the time until completion of all project activities. The financial aspects of a project are often overlooked. In most capital-intensive industries, however, the value of a project is much more important than its completion time. Traditionally, the value of a project is expressed using its net present value (NPV). The NPV of a project is obtained by discounting all cash flows incurred during the project lifetime. The most common goal of a scheduling procedure then is to schedule the project activities such that the expected NPV (or eNPV) of a project is maximized.

In a recent survey, Wiesemann and Kuhn (2015) not only highlight the importance of NPV over project completion time, but also stress the importance of stochastic project scheduling. In stochastic project scheduling, the cash flows and/or the activity durations are random variables, and hence, the project NPV itself is also a random variable. In this article, we study the stochastic NPV maximization problem (SNPV), and assume that activity durations are phase-type (PH) distributed. PH distributions are mixtures of exponential distributions that can be used to approximate any positive-valued distribution. As such, they not only allow to model activity durations in a general way, they also allow us to still exploit the properties of the exponential distribution. In accordance with most of the literature on the SNPV, we assume that cash flows are deterministic, and that there are no resources. For exponentially-distributed activity durations, Sobel et al. (2009) were the first to propose a generic formulation of the SNPV as a continuous-time Markov decision process. Creemers et al. (2010) extend the work of Sobel et al. (2009), and present a procedure that is the current state-of-the-art for solving the SNPV. Both Sobel et al. (2009) and Creemers et al. (2010) use the well-known continuous-time Markov chain (CTMC) of Kulkarni and Adlakha (1986). In this article, we take a different approach and use the new CTMC of Creemers (2016). Although the CTMC of Creemers (2016) is far more memory-efficient than the CTMC of Kulkarni and Adlakha (1986), it also allows activities to be preempted; an assumption that is not always desirable. In what follows, however, we prove that it is globally optimal not to preempt activities when solving the SNPV.

Our main contributions are: (1) we extend the approach of Creemers (2016), and demonstrate that it can be used to find globally optimal solutions for the SNPV, and (2) we significantly improve computational performance when compared to the current-state-of-the-art procedures for solving the SNPV.

The remainder of this article is structured as follows. Section 2 reviews the literature. Section 3 presents some basic definitions and provides a brief problem description. Section 4 defines the CTMC, and explains how a backward stochastic dynamic-programming (SDP) recursion is used to obtain the maximum eNPV of a project. Section 5 introduces the PH distributions that are used to model the activity durations. Section 6 discusses when to preempt activities, and why it is globally optimal to not preempt activities when solving the SNPV. Section 7 reports on the results of an elaborate computational experiment. Section 8 concludes.

2 Literature review

The idea of maximizing the NPV of a project was first introduced by Russell (1970) , who describes a nonlinear model where all parameters are deterministic. A few year later, Grinold (1972) showed that the nonlinear model of Russell can be transformed into an equivalent linear model that can be solved using a network simplex algorithm. In 2000, Neumann and Zimmermann extend Grinold's model to solve problems with generalized precedence relationships. The deterministic NPV maximization problem has also been studied by Elmaghraby and Herroelen (1990) and Schwindt and Zimmermann (2001), who develop efficient solution procedures. Heuristic procedures are available from Baroum and Patterson (1996) and Pinder and Marucheck (1996). Several extensions have been considered that allow for resource constraints, time-dependent cash flows, and multiple execution modes. For a review of the literature on the deterministic NPV maximization problem and its extensions, we refer the reader to Herroelen et al. (1997), Demeulemeester and Herroelen (2002), Hartmann and Briskorn (2010), Gu et al. (2015), and Wiesemann and Kuhn (2015).

If activity durations are exponentially distributed, Buss and Rosenblatt (1997) were the first to maximize the eNPV of a project while observing the impact of activity delay. The SNPV with exponentially-distributed activity durations itself, however, was only formally defined in 2009 by Sobel et al. (2009). Their work was continued by Creemers et al. (2010), who propose a memory-efficient procedure to solve the SNPV. The procedure of Creemers et al. (2010) has, among others, been used by Gutin et al. (2015) to study interdiction games, and has also been adapted for solving the stochastic resource-constrained project scheduling problem (SRCPSP) by Creemers (2015). Sobel et al. (2009) and Creemers (2015) both discuss the possibility to use PH distributions to model activity durations. The impact of PH distributions and activity duration variability are explored in Creemers et al. (2015b), who maximize the eNPV of modular projects under the assumption that activities can fail. Note that all aforementioned works use the CTMC of Kulkarni and Adlakha (1986) to model project networks.

Instead of using exponential distributions, Tavares et al. (1998) adopt lognormal activity durations and normally distributed cash flows to solve generic project scheduling problems. Discrete duration and cash flow distributions are used by Benati (2006), who presents a two-stage heuristic to solve the SNPV. Wiesemann et al. (2010) also use discrete distributions, and develop an exact branch-and-bound procedure. Resources and multiple execution modes have been considered in Özdamar and Dündar (1997) and Chen and Zhang (2012). Ke and Liu (2005) consider three stochastic models that allow to minimize: (1) the expected cost, (2) the cost while imposing chance constraints, and (3) the probability to overrun the budget.

As an alternative to probability distributions, fuzzy numbers are often used to represent activity durations and/or cash flows. The fuzzy NPV maximization problem has been introduced by Uçal and Kuchta (2011), who assume fuzzy cash flows and deterministic activity durations. Shavandi et al. (2012), on the other hand, assume fuzzy activity durations and deterministic cash flows. Ke and Liu (2007, 2010) extend their work of 2005 to also accommodate fuzzy activity durations.

3 Definitions and problem description

A project can be seen as a graph $G = (V, E)$, where $V = \{1, \dots, n\}$ is a set of nodes that represent project activities, and $E = \{(i, j) | i, j \in V\}$ is a set of arcs that represent precedence relationships. The start and the completion of a project are represented by dummy activities 1 and n, respectively. Each non-dummy activity $i : i \in V \setminus \{1, n\}$ has a random duration $\tilde{p}_i$ with expectation μ_i and variance σ_i^2. In addition, $\tilde{\mathbf{p}} = \{\tilde{p}_2, \tilde{p}_3, \dots, \tilde{p}_{n-1}\}$ denotes the vector of random variables $\tilde{p}_i$, and $\mathbf{p} = \{p_2, p_3, \dots, p_{n-1}\}$ is the vector of random variates (or realizations) of $\tilde{\mathbf{p}}$, where p_i is a random variate of $\tilde{p}_i$.

Because activity durations are uncertain, activity starting times cannot be defined at the start of the project. Instead, they are determined during project execution using a policy. Most of the literature on stochastic project scheduling adopts simple list policies that execute activities in the order of a list (see, e.g., Golenko-Ginzburg & Gonik, 1997; Tsai & Gemmill, 1998; Ballestín & Leus, 2009; Ashtiani et al., 2011; Rostami et al., 2017). In this article, on the other hand, we adopt elementary policies; a more general class of policies that allows decisions to be made at the start of the project and at the end of activities. List policies are a subset of the class of elementary policies, and, in turn, elementary policies are a subset of the class of all policies (refer to Rostami et al., 2017 for a hierarchy of the different policy classes). Even though elementary policies are a subset of the set of all policies, they have been shown to be globally optimal when solving the SNPV if activity durations are exponentially distributed (Sobel et al., 2009).

A policy can be seen as a set of decision rules that define 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 abandonment of the project or the start/interruption of a 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/at time t). When executing a policy, activity starting times become known gradually (i.e., a schedule is constructed as time progresses). As a result, a policy Π may be interpreted as a function that maps realizations of activity durations $\mathbf{p}$ to vectors of feasible starting times $\mathbf{S}(\mathbf{p},\Pi) = \{S_1(\mathbf{p},\Pi), S_2(\mathbf{p},\Pi), \ldots, S_n(\mathbf{p},\Pi)\}$, where $S_1(\mathbf{p},\Pi) = 0$ and $S_n(\mathbf{p},\Pi) = \max\limits_{i \in V \setminus \{1,n\}} S_i(\mathbf{p},\Pi) + p_i$. Refer to Igelmund and Radermacher (1983), Möhring (2000), and Stork (2001) for more details.

Without loss of generality, we assume that a cash flow c_i is incurred at the start of activity i, where c_1 represents the initial outlay, and c_n represents the projet payoff. We use continuous discounting to determine the eNPV of a cash flow c_i:

$$v_i = E\left(c_i e^{-rS_i(\mathbf{p},\Pi)}\right), \tag{1}$$

where r is the discount rate, and $E(\cdot)$ is the expectation operator with respect to $\mathbf{p}$. The eNPV of the project is:

$$v = \sum_{i \in V} v_i. \tag{2}$$

The optimal policy $\Pi^\star$ selects activities such that v is maximized.

4 CTMC and SDP recursion

In this section, we assume that activity durations are exponentially distributed (later on, in Section 5, we introduce PH-distributed activity durations). If activity durations are exponentially distributed, a project can be seen as a Markovian PERT network. Markovian PERT networks were first studied by Kulkarni and Adlakha (1986), who use a CTMC to obtain the exact distribution of the earliest completion time of a project. In the CTMC of Kulkarni and Adlakha (1986), the state of the system is defined by three sets: the set of idle activities I, the set of ongoing activities O, and the set of finished activities F. Because activities are either idle, ongoing, or finished, the size of the state space of the CTMC has upper bound 3^n. Even for small n, storing all states in memory is impossible. Most states, however, do not satisfy precedence constraints. In order to reduce memory requirements, a strict partitioning of the state space is required. Creemers et al. (2010) introduce the use of uniformly directed cuts (UDCs) to structure the state space. Afterwards, their approach was adopted by, among others, Wei et al. (2013), who solve a sequential testing problem, and Coolen et al. (2014), who solve a single-machine scheduling problem with modular projects. Although UDCs allow to generate only the feasible states, the identification of UDCs themselves is NP-hard (Shier and Whited 1986).

Until recently, all of the work on Markovian PERT networks uses the well-known CTMC of Kulkarni and Adlakha (1986). In 2016, however, Creemers has introduced a new CTMC that, in contrast to the CTMC of Kulkarni and Adlakha (1986), only keeps track of the set of finished activities. As a result, the size of the state space has upper bound 2^n. In addition, Creemers (2016) no longer uses UDCs to structure the state space. Instead, he uses a set of two ordered arrays that not only reduces the computational effort required to generate/search the state space, but also reduces the number of states that are stored in memory at any one time. Creemers (2016) uses this new approach to tackle the preemptive stochastic resource-constrained project scheduling problem (PSRCPSP), and is able to easily outperform existing optimal procedures for similar scheduling problems.

In this article, we use the CTMC and state-space structure of Creemers (2016). This requires us to adopt a different approach than the one that is used in works that rely on the CTMC of Kulkarni and Adlakha (1986). In these latter works, the optimal policy tries to determine the optimal set of activities to start in each state. In this work, however, we only keep track of the set of finished activities, and as such have no idea of which activities are idle/ongoing. In other words, we cannot determine the optimal set of activities to start (some of them might already be ongoing). Instead, we first determine the set of activities that are potentially ongoing, and next, the optimal policy selects the optimal set of ongoing activities. More formally, let $F(t)$ and $H(F,t)$ denote the set of finished and potentially ongoing activities at time t, respectively. An activity i is potentially ongoing at time t if: (1) i is not in $F(t)$ and (2) $j \in F(t)$ for all j for which $(j,i) \in E$. From $H(F,t)$, policy Π selects the set of ongoing activities. The start and completion of the project are defined as $F(0) = \emptyset$ and $F(t) = V$ for all $t \geq \omega$, where ω is the completion time of the project. Without loss of generality, we omit index t when referring to sets $F(t)$ and $H(F,t)$.

The state of the system is represented by the set of finished activities (F). Upon entry of state $(F) : F \neq V$, policy Π determines the set of ongoing activities $O(\Pi,F) \subseteq H(F)$. The optimal policy $\Pi^\star$ selects $O(\Pi^\star,F)$ from $H(F)$ such that $G(\Pi^\star,F)$ is maximized, where $G(\Pi,F)$ is the value function that returns the eNPV of the project upon entry of state (F) if policy Π is adopted. Given a set of ongoing activities O, the time until the first completion of an activity $i : i \in O$ is exponentially distributed with expected value $(\sum_{i\in O}\lambda_i)^{-1}$. The probability that activity $i : i \in O$ finished first equals $\lambda_i(\sum_{j\in O}\lambda_j)^{-1}$. Therefore, if policy Π is adopted, the eNPV of the project upon entry of state (F) equals:

$$G(\Pi,F) = \frac{\sum\limits_{j\in O(\Pi,F)}\lambda_j}{r + \sum\limits_{j\in O(\Pi,F)}\lambda_j} \sum_{i\in O(\Pi,F)} \frac{\lambda_i}{\sum\limits_{j\in O(\Pi,F)}\lambda_j} \left(G\left(\Pi, F\cup\{i\}\right) + \sum_{j\in O(\Pi,F\cup\{i\})\setminus O(\Pi,F)} c_j \right), \quad (3)$$

where $\sum_{j\in O(\Pi,F\cup\{i\})\setminus O(\Pi,F)} c_j$ is the cash flow that is incurred when starting activities for the first time upon entry of state $(F\cup\{i\})$. The optimal subset of ongoing activities is given by:

$$O(\Pi^\star,F) = \underset{O\subseteq H(F)}{\arg\min} \frac{\sum\limits_{j\in O}\lambda_j}{r + \sum\limits_{j\in O}\lambda_j} \sum_{i\in O} \frac{\lambda_i}{\sum\limits_{j\in O}\lambda_j} \left(G\left(\Pi^\star, F\cup\{i\}\right) + \sum_{j\in O(\Pi^\star,F\cup\{i\})\setminus O} c_j \right). \quad (4)$$

Finding the optimal set of ongoing activities requires us to enumerate all subsets of $H(F)$. Note, however, that several heuristics may be devised in order to determine a "good" set of ongoing activities. In addition, note that, if no set $O \subseteq H(F)$ can be found that results in a positive eNPV, $O(\Pi^\star, F) = \emptyset$ if project abandonment is allowed.

In order to structure the state space of the CTMC, we adopt the approach of Creemers (2016), and use a set of two ordered arrays $\mathbf{X}_i$ and $\mathbf{X}_{i+1}$. Array $\mathbf{X}_i$ contains all feasible states in which i activities are finished. From a state $(F) \in \mathbf{X}_i$ transitions are only possible towards states in $\mathbf{X}_{i+1}$. As a result, it suffices to keep only two arrays in memory. We use a backward SDP-recursion to determine the maximum eNPV of a project. The recursion starts in state $(F) = V$, and completes upon reaching state $(F) = \emptyset$. For each state $(F) \in \mathbf{X}_i$, we use Eq. (3) to determine $G(\Pi^\star, F)$, with $G(\Pi^\star, V) = c_n$. After all states in $\mathbf{X}_i$ have been processed, array $\mathbf{X}_{i+1}$ is no longer needed, and it is used to store the value functions of all states in which $i-1$ activities have finished (i.e., $\mathbf{X}_{i+1}$ becomes $\mathbf{X}_{i-1}$). Eventually, we obtain $G(\Pi^\star, \emptyset)$, the value function of state $(F) = \emptyset$, and have determined the maximum eNPV of the project.

5 PH-distributed activity durations

It is not always realistic to assume that activity durations are exponentially distributed. To overcome this limitation, we use phase-type (PH) distributions. PH distributions are mixtures of exponential distributions that can be used to approximate any positive-valued distribution with arbitrary precision (Neuts, 1981; Osogami, 2005). Using PH distributions, we can approximate the "true" duration distribution of an activity. The more accurate the approximation, the more complex the PH distribution tends to become. For instance, we may need a complex PH distribution if we want to match the first four moments of the duration distribution. For matching the first two moments, however, we only require very simple PH distributions. We use these simple PH distributions to match the first two moments of the true duration distribution. Although this implies that the mean and the variance of the PH distribution match those of the true distribution, it does not necessarily mean that higher moments (i.e., skewness, kurtosis, ...) are also matched. In practice, however, the true duration distribution is often unknown. In most cases, the mean and variance are the only information available. Therefore, matching only the first two moments is sufficient/makes sense from a practical point of view.

Let $\nu_i = \sigma_i^2 \mu_i^{-2}$ denote the squared coefficient of variation (SCV) of the duration of an activity i. If $\nu_i = 1$, the duration distribution of activity i can be approximated by an exponential distribution with rate parameter $\lambda_i = \mu_i^{-1}$. If $\nu_i < 1$, on the other hand, we use a hypo-exponential distribution (a generalization of the Erlang distribution) to approximate the duration distribution of an activity i. The hypo-exponential distribution has $z_i = \lceil \nu_i^{-1} \rceil$ phases, and the first $z_i - 1$ phases have exponential duration with rate parameter:

$$\lambda_{i,1} = \lambda_{i,2} = \ldots = \lambda_{i,z_i-1} = \frac{(z_i - 1) - \sqrt{(z_i - 1)(z_i \nu_i - 1)}}{\mu_i (1 - \nu_i)}. \tag{5}$$

The last phase has exponential duration with rate parameter:

$$\lambda_{i,z_i} = \frac{1 + \sqrt{(z_i - 1)(z_i\nu_i - 1)}}{\mu_i(1 - z_i\nu_i + \nu_i)}. \tag{6}$$

Because an SCV of 1 is already seen as highly variable (see, e.g., Ballestín & Leus, 2009; Ashtiani et al., 2011; Rostami et al., 2017), we do not consider the case where $\nu_i > 1$.

Because PH distributions are mixtures of exponential distributions, a project network with PH-distributed activity durations can be transformed into a Markovian PERT network (see Creemers, 2015, for more details). As a result, we can once more use the SDP recursion introduced in Section 4 to obtain the maximum eNPV of the project.

6 To preempt or not to preempt?

Although the CTMC of Creemers (2016) is able to drastically reduce memory requirements, it also has one limitation: it allows activities to be preempted. The SNPV, however, does not allow preemption. As such, the CTMC of Creemers (2016) is in fact not fit to solve the SNPV. In this section, we justify the use of the CTMC of Creemers (2016) by proving that it is globally optimal not to preempt activities when solving the SNPV. Moreover, this proof holds even if activities have durations that are not PH distributed.

Lemma 1. *If at time t there is an eligible activity i, and if a cash flow $c_i = 0$ is incurred at the start of activity i, then there exists an optimal continuation where activity i is put in progress at time t.*

Proof. The proof is obvious. There is no reason to postpone the start of an activity if no cash flow is incurred at the start of that activity. □

Theorem 1. *If cash flows are incurred at the start of an activity, it is globally optimal to not preempt activities.*

Proof. Let O_t be the non-empty set of ongoing activities at time t. If we decide to preempt activity $i : i \in O_t$ at time t, the remainder of activity i joins the set of eligible activities. Because cash flow c_i has already been incurred at some time $t' \leq t$, a zero cash flow is incurred upon the start of the remainder of activity i. From Lemma 1, however, it follows that it is globally optimal to start any eligible activity that has a zero cash flow as early as possible. As a result, it is globally optimal to start the remainder of activity i at time t (i.e., at the same time that activity i was preempted). In other words, it is globally optimal to not preempt activity i. □

7 Computational results

Even though the procedure of Creemers et al. (2010) is still the current state-of-the-art for solving the SNPV, their procedure has significantly been improved by Creemers (2015), who uses it to solve the SRCPSP. In order to take these improvements into account, we adapt the procedure of Creemers (2015) such that it can be used to also solve the SNPV. Note that both approaches still rely on the CTMC of Kulkarni and Adlakha (1986), and that they use UDCs to structure the state space.

To test the performance of their procedure, Creemers et al. (2010) use a data set of 1,080 instances. They use RANGEN (Demeulemeester et al., 2003) to generate 30 projects for each combination of project size (ranging from 12 to 122 activities) and order strength (OS), where OS is a measure that reflects the density of the project network. Creemers et al. (2010) consider OS equal to 0.4, 0.6, or 0.8. In what follows, we use the same data set to compare the performance of our approach with the state-of-the-art procedure of Creemers et al. (2010) and the adapted (and unpublished) procedure of Creemers (2015). To allow for a fair comparison, we perform all tests on the same system: an AMD Phenom II 3.4 Ghz desktop computer with 32GB of RAM.

Tables 1-3 compare the CPU times, the size of the state space, and the improvement factors of the different approaches for different values of n and OS. Note that we do not report on all combinations of n and OS because we can only compare the performance for those instances that can be solved by Creemers et al. (2010). From Tables 1-3, it is clear that major improvements have been made with respect to CPU times when we compare the procedures of Creemers et al. (2010) and Creemers (2015). As also explained in Creemers (2015), however, the main bottleneck is the size of the state space (i.e., the memory requirement) rather than the CPU time. Because both procedures use the same CTMC, the number of states remains unchanged. This also clearly illustrates the importance of the CTMC of Creemers (2016): it allows to drastically reduce the size of the state space (and hence memory requirements). Not only is the new approach significantly faster, it also allows us to remove the bottleneck: memory is no longer a constraint. For instance, the most complex instance that can be solved by Creemers et al. (2010) has a state-space size of 867,589,281 states, and requires 318,464 seconds to solve (1,687 seconds when using the adapted procedure of 2015). To solve the same instance, we require only 1,846,012 states, and a computation time of 149.9 seconds.

Over all instances, our new approach improves computational efficiency by a factor of 600, and reduces memory requirements by a factor of 321. Table 3 also makes clear that the difference in performance becomes bigger as the instances become more complex. This can be explained by the fact that the maximum size of the state space is 3^n for the procedures of Creemers et al. (2010) and Creemers (2015), and only 2^n for the new approach.

We conclude that our approach easily outperforms the procedures of Creemers et al. (2010) and Creemers (2015), and that it is the new state-of-the-art for solving the SNPV.

Table 4 allows to further illustrate the computational performance of our approach. From Table 4, it is apparent that memory requirements are never the problem. With a maximum state-space size of 502,600,920 states, we are still well below the maximum size used by Creemers et al. (2010). If we look at CPU times, however, we see that the more complex instances take a very long time to solve. We conclude that the bottleneck has shifted from memory to computation time.

Table 1: Comparison of computation times (in seconds) of the different approaches

			2010		2015		2017	
n	OS	Solved	avg	max	avg	max	avg	max
12	0.8	30	0.002	0.015	0	0	0	0
12	0.6	30	0.002	0.016	0	0	0	0
12	0.4	30	0.004	0.016	0.001	0.016	0	0
22	0.8	30	0.006	0.016	0	0	0	0
22	0.6	30	0.015	0.047	0.001	0.016	0.002	0.015
22	0.4	30	0.463	1.841	0.035	0.109	0.006	0.016
32	0.8	30	0.011	0.016	0.001	0.016	0.003	0.016
32	0.6	30	0.338	0.967	0.030	0.063	0.009	0.016
32	0.4	30	26.93	161.4	1.880	9.800	0.180	0.748
42	0.8	30	0.037	0.063	0.005	0.016	0.004	0.016
42	0.6	30	6.633	29.64	0.660	1.950	0.068	0.172
42	0.4	29	2,338	11,314	92.18	318.9	6.687	22.75
52	0.8	30	0.162	0.390	0.021	0.047	0.010	0.016
52	0.6	30	100.3	497.6	6.001	29.76	0.593	1.888
52	0.4	4	52,268	91,628	1,048	1,335	82.09	117.2
62	0.8	30	0.763	2.996	0.091	0.234	0.019	0.031
62	0.6	30	2,210	13,663	88.74	410.2	6.197	29.06
72	0.8	30	3.219	8.955	0.357	0.749	0.054	0.093
72	0.6	22	17,496	64,805	504.6	1,074	34.18	73.99
82	0.8	30	10.86	41.59	1.137	3.651	0.143	0.359
82	0.6	9	106,033	318,467	1,578	1,772	120.2	149.9
92	0.8	30	50.71	308.9	4.521	17.86	0.407	1.310
102	0.8	30	171.6	900.9	13.79	47.36	1.122	4.118
112	0.8	30	1,194	11,376	57.32	337.5	3.906	19.63
122	0.8	30	12,790	70,180	360.7	1,123	26.22	87.09

Table 2: Comparison of state-space size (in 1,000 states) required by the different approaches

			2010		2015		2017	
n	OS	Solved	avg	max	avg	max	avg	max
12	0.8	30	0.071	0.105	0.071	0.105	0.022	0.025
12	0.6	30	0.206	0.333	0.206	0.333	0.041	0.050
12	0.4	30	0.695	2.361	0.695	2.361	0.084	0.148
22	0.8	30	0.484	0.953	0.484	0.953	0.088	0.112
22	0.6	30	4.006	7.673	4.006	7.673	0.330	0.462
22	0.4	30	55.02	153.4	55.02	153.4	1.620	2.760
32	0.8	30	1.995	3.233	1.995	3.233	0.254	0.317
32	0.6	30	49.39	84.84	49.39	84.84	1.898	2.394
32	0.4	30	1,560	5,967	1,560	5,967	17.10	32.60
42	0.8	30	7.860	11.95	7.860	11.95	0.662	0.794
42	0.6	30	534.0	1543	534.0	1,543	9.480	16.58
42	0.4	29	47,073	146,560	47,073	146,560	171.7	316.9
52	0.8	30	26.67	53.48	26.67	53.48	1.544	2.194
52	0.6	30	4,346	13,894	4,346	13,894	40.38	67.98
52	0.4	4	526,020	737,048	526,020	737,048	1,055	1,349
62	0.8	30	92.00	236.9	92.00	236.9	3.564	5.362
62	0.6	30	42,279	165,103	42,279	165,103	175.3	365.8
72	0.8	30	286.8	605.6	286.8	605.6	7.754	11.39
72	0.6	22	216,028	426,644	216,028	426,644	593.1	860.1
82	0.8	30	829.7	2,278	829.7	2,278	16.36	27.89
82	0.6	9	733,449	867,589	733,449	867,589	1,585	1,846
92	0.8	30	2,596	9,322	2,596	9,322	34.06	55.88
102	0.8	30	6,868	22,963	6,868	22,963	66.67	113.2
112	0.8	30	24,236	117,261	24,236	117,261	146.9	308.5
122	0.8	30	146,639	461,146	146,639	461,146	515.8	916.4

Table 3: Average computational improvement factor for different combinations of approaches

			CPU times			State-space size		
n	OS	Solved	2010 vs. 2017	2010 vs. 2015	2015 vs. 2017	2010 vs. 2017	2010 vs. 2015	2015 vs. 2017
12	0.8	30	NA	NA	NA	3.280	0	3.280
12	0.6	30	NA	NA	NA	5.048	0	5.048
12	0.4	30	NA	7.750	NA	8.253	0	8.253
22	0.8	30	NA	NA	NA	5.522	0	5.522
22	0.6	30	10.04	14.13	0.711	12.14	0	12.14
22	0.4	30	75.57	13.36	5.658	33.96	0	33.96
32	0.8	30	4.247	10.55	0.403	7.854	0	7.854
32	0.6	30	36.74	11.20	3.279	26.03	0	26.03
32	0.4	30	149.2	14.32	10.42	91.27	0	91.27
42	0.8	30	9.082	7.858	1.156	11.87	0	11.87
42	0.6	30	97.50	10.05	9.699	56.33	0	56.33
42	0.4	29	349.7	25.36	13.79	274.1	0	274.1
52	0.8	30	15.87	7.831	2.026	17.27	0	17.27
52	0.6	30	169.2	16.72	10.12	107.6	0	107.6
52	0.4	4	636.7	49.87	12.77	498.6	0	498.6
62	0.8	30	39.28	8.429	4.660	25.81	0	25.81
62	0.6	30	356.7	24.91	14.32	241.2	0	241.2
72	0.8	30	59.22	9.021	6.564	36.99	0	36.99
72	0.6	22	511.9	34.67	14.76	364.3	0	364.3
82	0.8	30	76.03	9.548	7.963	50.71	0	50.71
82	0.6	9	881.8	67.20	13.12	462.7	0	462.7
92	0.8	30	124.7	11.21	11.12	76.24	0	76.24
102	0.8	30	152.9	12.44	12.29	103.0	0	103.0
112	0.8	30	305.7	20.84	14.67	165.0	0	165.0
122	0.8	30	487.8	35.46	13.76	284.3	0	284.3

Table 4: Average computation time (in seconds) and state-space size (in 1,000 states) required by our approach

			CPU times		State-space size	
n	OS	Solved	avg	max	avg	max
10	0.8	30	0	0	0.022	0.025
10	0.6	30	0	0	0.041	0.050
10	0.4	30	0	0	0.084	0.148
20	0.8	30	0	0	0.088	0.112
20	0.6	30	0.002	0.015	0.330	0.462
20	0.4	30	0.006	0.016	1.620	2.760
30	0.8	30	0.003	0.016	0.254	0.317
30	0.6	30	0.009	0.016	1.898	2.394
30	0.4	30	0.180	0.748	17.10	32.60
40	0.8	30	0.004	0.016	0.662	0.794
40	0.6	30	0.068	0.172	9.480	16.58
40	0.4	30	11.85	161.5	193.8	834.8
50	0.8	30	0.010	0.016	1.544	2.194
50	0.6	30	0.593	1.888	40.38	67.98
50	0.4	30	255.1	1,111	1,660	3,901
60	0.8	30	0.019	0.031	3.564	5.362
60	0.6	30	6.197	29.06	175.3	365.8
60	0.4	30	8,454	62,555	13,791	38,029
70	0.8	30	0.054	0.093	7.754	11.39
70	0.6	30	58.57	261.6	727.9	1,749
70	0.4	30	131,386	750,675	102,937	502,601

8 Conclusion

In this article, we consider projects with stochastic activity durations that are modeled using PH distributions. Intermediate cash flows are incurred during the execution of the project, and a payoff is obtained upon completion of all project activities. For such projects, we find globally optimal policies that maximize the eNPV.

We build on the work of Creemers et al. (2010), and use the CTMC of Creemers (2016) to develop a new procedure to solve the SNPV. Although the CTMC of Creemers (2016) is far more memory-efficient than the well-known CTMC of Kulkarni and Adlakha (1986), it has one limitation: it allows activities to be preempted. The SNPV, on the other hand, does not allow for preemption. In other words, the CTMC of Creemers (2016) is in fact not fit to solve the SNPV. We prove, however, that it is globally optimal to not preempt activities when solving the SNPV. Moreover, this proof holds even if activities have durations that are not PH distributed.

We perform a computational experiment to: (1) assess the computational efficiency of our approach and (2) compare our approach with the current state-of-the-art procedures. From the computational experiment, it is clear that our new approach significantly outperforms the current state-of-the-art procedure of Creemers et al. (2010) and the adapted/unpublished procedure of Creemers (2015). On average, we reduce memory requirements by a factor of 321, and are able to improve computational efficiency by a factor of 600. In addition, the computational experiment also reveals that the bottleneck has shifted from memory to computation time. Large/complex instances can be solved to optimality, albeit at a significant computational cost.

Fortunately, our procedure can easily be transformed into a heuristic that requires much less computation time. Our approach requires to determine the set of ongoing activities in each state of the system. The optimal set of ongoing activities is found using full enumeration. Instead of enumerating all possible sets of ongoing activities, a heuristic can be used to quickly determine a "good" set of ongoing activities. Another direction for future research is to extend our procedure to also include activity failures, resources, and/or multiple execution modes.

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

[3] Baroum, S. M., & Patterson, J. H. (1996). The development of cash flow weight procedures for maximizing the net present value of a project. *Journal of Operations Management*, *14(3)*, 209–227.

[4] Benati, S. (2006). An optimization model for stochastic project networks with cash flows. *Computational Management Science*, *3(4)*, 271–284.

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

[6] Chen, W-N., & Zhang, J. (2012). Scheduling multi-mode projects under uncertainty to optimize cash flows: A Monte Carlo ant colony system approach. *Journal of Computer Science and Technology*, *27(5)*, 950–965.

[7] Coolen, K., Wei, W., Talla Nobibon, F., & Leus, R. (2014). Scheduling modular projects on a bottleneck resource. *Journal of Scheduling*, *17(1)*, 67-85.

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

[9] Creemers, S. (2015). Minimizing the expected makespan of a project with stochastic activity durations under resource constraints. *Journal of Scheduling*, *18(3)*, 263–273.

[10] Creemers, S., Leus, R., & De Reyck, B. (2015b). Project planning with alternative technologies in uncertain environments. *European Journal of Operational Research*, *242(2)*, 465–476.

[11] Creemers, S. (2016). *The preemptive stochastic resource-constrained project scheduling problem: an efficient globally optimal solution procedure*, Working paper KBI_1626. KU Leuven, Faculty of Economics and Business, Department of Decision Sciences and Information Management.

[12] Demeulemeester, E. L., & Herroelen, W. S. (2002). *Project Scheduling A Research Handbook.* Kluwer Academic Publishers.

[13] Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). A random network generator for activity-on-the-node networks. *Journal of Scheduling*, *6(1)*, 17-38.

[14] Elmaghraby, S. E., & Herroelen, W. S. (1990). The scheduling of activities to maximize the net present value of projects. *European Journal of Operational Research. 49(1)*, 35–49.

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

[16] Grinold, R. C. (1972). The payment scheduling problem. *Naval Research Logistics Quarterly*, *19(1)*, 123–136.

[17] Gu, H., Schutt, A., Stuckey, P. J., Wallace, M. G., & Chu, G. (2015). Exact and heuristic methods for the resource-constrained net present value problem. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol. 1.* New York: Springer.

[18] Gutin, E., Kuhn, D., & Wiesemann, W. (2015). Interdiction Games on Markovian PERT Networks. *Management Science*, *61(5)*, 999–1017.

[19] Hartmann, S., & Briskorn, D. (2010). A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, *207(1)*, 1–14.

[20] Herroelen, W. S., Van Dommelen, P., & Demeulemeester, E.L. (1997). Project network models with discounted cash flows a guided tour through recent developments. *European Journal of Operational Research*, *100(1)*, 97–121.

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

[22] Ke, H., & Liu, B. (2005). Project scheduling problem with stochastic activity duration times. *Applied Mathematics and Computation*, *168(1)*, 342–353.

[23] Ke, H., & Liu, B. (2007). Project scheduling problem with mixed uncertainty of randomness and fuzziness. *European Journal of Operational Research*, *183(1)*, 135–147.

[24] Ke, H., & Liu, B. (2010). Fuzzy project scheduling problem and its hybrid intelligent algorithm. *Applied Mathematical Modelling*, *34(2)*, 301–308.

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

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

[27] Neumann, K., & Zimmermann, J. (2000). Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, *127(2)*, 425–443.

[28] Neuts, M. F. (1981). *Matrix-Geometric Solutions in Stochastic Models.* Johns Hopkins University Press.

[29] Osogami, T. (2005). *Analysis of multiserver systems via dimensionality reduction of Markov chains*. Pittsburgh: Carnegie Mellon University Ph. D. dissertation.

[30] Özdamar, L., & Dündar, H. (1997). A flexible heuristic for a multi-mode capital constrained project scheduling problem with probabilistic cash inflows. *Computers & Operations Research*, *24(12)*, 1187–1200.

[31] Pinder, J. P., & Marucheck, A. S. (1996). Using discounted cash flow heuristics to improve project net present value. *Journal of Operations Management*, *14(3)*, 229–240.

[32] Rostami, S., Creemers, S., & Leus, R. (2017). New strategies for stochastic resource-constrained project scheduling. *Journal of Scheduling.* doi:10.1007/s10951-016-0505-x.

[33] Russell, A. H. (1970). Cash Flows in Networks. *Management Science*, *16(5)*, 357–373.

[34] Shavandi, H., Najafi, A. A., & Moslehirad, A. (2012). Fuzzy project scheduling with discounted cash flows. *Economic Computation & Economic Cybernetics Studies*, *46(1)*, 219–232.

[35] Schwindt, C., & Zimmermann, J. (2001). A steepest ascent approach to maximizing the net present value of projects. *Mathematical Methods of Operations Research*, *53(3)*, 435–450.

[36] Shier, D. R., & Whited, D. E. (1986). Iterative algorithms for generating minimal cutsets in directed graphs. *Networks*, *16(2)*, 133–147.

[37] 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.

[38] Stork, F. (2001). *Stochastic Resource-Constrained Project Scheduling.* Berlin: Technische Universität Ph. D. dissertation.

[39] Tavares, L. V., Ferreira, J. A. A., & Coelho, J. S. (1998). On the optimal management of project risk. *European Journal of Operational Research*, *107(2)*, 451–469.

[40] 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.

[41] Uçal, İ., & Kuchta, D. (2011). Project scheduling to maximize fuzzy net present value. In *Proceedings of the world congress on engineering* (pp. 1184–1189).

[42] Wei, W., Coolen, K., & Leus, R. (2013). Sequential testing policies for complex systems under precedence constraints. *Expert Systems Applications*, *40(2)*, 611-620.

[43] Wiesemann, W., Kuhn, D., & Rustem, B. (2010). Maximizing the net present value of a project under uncertainty. *European Journal of Operational Research*, *202(2)*, 356–367.

[44] Wiesemann, W., & Kuhn, D. (2015). The stochastic time-constrained net present value problem. In C. Schwindt, & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling Vol. 2.* New York: Springer.