The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Optimal Approximation of Random Variables for Estimating the Probability of Meeting a Plan Deadline

Liat Cohen,[1] **Tal Grinshpoun,**[2] **Gera Weiss**[1]
[1]Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
[2]Department of Industrial Engineering and Management
Ariel University, Ariel 40700, Israel

Abstract

In planning algorithms and in other domains, there is often a need to run long computations that involve summations, maximizations and other operations on random variables, and to store intermediate results. In this paper, as a main motivating example, we elaborate on the case of estimating probabilities of meeting deadlines in hierarchical plans. A source of computational complexity, often neglected in the analysis of such algorithms, is that the support of the variables needed as intermediate results may grow exponentially along the computation. Therefore, to avoid exponential memory and time complexities, we need to trim these variables. This is similar, in a sense, to rounding intermediate results in numerical computations. Of course, to maintain the quality of algorithms, the trimming procedure should be efficient and it must maintain accuracy as much as possible. In this paper, we propose an optimal trimming algorithm with polynomial time and memory complexities for the purpose of estimating probabilities of deadlines in plans. More specifically, we show that our algorithm, given the needed size of the representation of the variable, provides the best possible approximation, where approximation accuracy is considered with a measure that fits the goal of estimating deadline meeting probabilities.

1 Introduction

Various stochastic problems in AI, and more specifically, in the decision making and planning literature, involve deadlines (Herroelen and Leus 2005; Beck and Wilson 2007; Fu, Varakantham, and Lau 2010; Buyya, Garg, and Calheiros 2011; Cohen, Shimony, and Weiss 2015). In these problems, the goal is to estimate the probability that a random variable gets a value that is below a given threshold (a deadline). If the random variable is X and the deadline is T, this amounts to evaluating $F_X(T)$ where F_X is the cumulative distribution function (CDF) of X. While this is straightforward when the distribution of X is given explicitly, it may be computationally hard when X is specified implicitly.

To demonstrate the type of computational complexities that we focus on in this paper, we briefly describe an example from the hierarchical planning domain. Assume that we are given a plan for the operation of a robot described as a tree whose leaves represent tasks with uncertain durations and whose nodes specify combinations of their sub-trees, that represent, recursively, composite tasks, either sequentially or in parallel. Consider the problem of estimating the probability that the plan makespan does not exceed a given deadline. Algorithms for solving such problems often apply operations like addition, multiplication and maximization, over random variables, see, e.g., (Shperberg, Shimony, and Felner 2017). In many cases the analysis found in the literature of the complexity of such algorithms focuses on the complexity stemming from the dependency graph and ignores the complexity that stems from the growth of the support, see, e.g., (Hagstrom 1988). However, when we look closer, we see that this complexity is not always negligible. For example, the size of the support of $X+Y$ may be a product of the sizes of the supports of X and of Y, i.e., the size of a standard representation of the variables, by a table that assigns a probability to each value in the support, may grow exponentially with additions. This may have a dramatic effect on both memory and time complexity of algorithms.

While there may be other solutions to the problem, e.g., by choosing different representations of the variables, our approach in this paper is to compute approximations of the random variables: Given a random variable X, we propose a way to compute a random variable X' with $\text{support}(X') < \text{support}(X)$, such that if we replace X with X' in the remainder of the computation, we get a result that is close to the one that we would have gotten if we used X. In this paper we use the notation $\text{support}(X)$ to denote the set of all the realizations of X that have a nonzero probability of being observed. The size of the support is proportional to the size of the table that people usually use as a data structure to represent discrete random variables.

The above idea is similar to the very common practice of rounding intermediate numerical results in long computations. This is a must when dealing with irrational numbers, e.g., $\sqrt{2} \approx 1.414$, and is also common for saving time and memory, e.g., rounding like $\$24.456 \approx \$24.46$ in financial computations.

While in the context of real numbers the notion of approximation, i.e., the distance (or rounding error) between two numbers, is clear, it is not so when dealing with random variables. There are various different distances known in the literature such as the Kolmogorov distance (Lilliefors 1967) and the Wasserstein distance (Vallender 1974). The

choice of a specific metric to use depends on the application at hand. In this work, motivated by the problem of estimating the probability of meeting deadlines, we focus on the Kolmogorov distance $d_K(X, X') = \sup_t |F_X(t) - F_{X'}(t)|$ where F_X and $F_{X'}$ are the CDFs of X and X', respectively.

The connection between the Kolmogorov distance and the deadline problem is clear – since $F_X(T)$ gives the probability of meeting the deadline T, the distance between F_X and $F_{X'}$ measures how similar are X and X' in terms of estimating the probabilities of meeting deadlines. This, however, is not enough for the applications that we are interested in since we need to be conservative in our estimations, i.e., overestimations of the probabilities of meeting deadlines are allowed, not underestimations. To this end, we adopt a one-sided version of the Kolmogorov distance. Specifically, we say that X' is an ε-approximation of X, denoted by $X \preceq_\varepsilon X'$, if $d_K(X, X') \leq \varepsilon$ and if, in addition, $F_X(t) \leq F_{X'}(t)$ for all t.

A similar notion of one-sided Kolmogorov approximation was also proposed in the work of Cohen, Shimony and Weiss (2015), where a polynomial-time (additive error) approximation scheme for computing the probability of meeting a deadline for task trees was suggested. Since the deadline problem of the complete task tree is NP-hard, they suggested an approximation algorithm. A key component in the suggested algorithm was a procedure for "trimming" the support of random variables. The Trim operator proposed in (Cohen, Shimony, and Weiss 2015) gets as input a random variable, X, and an error bound, ε, and returns as output a new random variable, X', such that $X \prec_\varepsilon X$. This is similar to what we do in this paper. However, the Trim operator is not optimal in the sense that there may exist a different random variable X'' with the same support size as X' such that $X'' \prec_{\varepsilon'} X$ and $\varepsilon' < \varepsilon$. In this paper we show that it is possible to find an optimal approximation in polynomial time. Note that there is still a trade-off of accuracy and time since Trim can be computed in linear time.

The present study is also a continuation of the work of Pavlikov and Uryasev (2016), where a procedure to produce a random variable X' that optimally approximates a random variable X is presented. Their approximation scheme, achieved using convex and linear programming, is designed for a different notion of distance (called CVaR). The new contribution of the present work in this context is that our method is direct, not using linear or convex programming, thus allowing tighter analysis of time and memory complexity. Also, our method is designed to approximate by the one-sided Kolmogorov error discussed above.

The rest of the paper is organized as follows. Section 2 describes the problem of task trees with deadlines that motivates this work. The main technical contribution is in Section 3, which includes the problem statement of optimally approximating random variables, followed by the description and properties of an algorithm that solves the problem at hand. Section 4 presents an empirical evaluation that demonstrates the quality of the proposed algorithm on the task trees domain, as well as additional experiments regarding the solution quality and run-time performance. A discussion and related work (Section 5) concludes the paper.

2 A Motivating Example: Task Trees with Deadlines

Hierarchical planning is a well-established field in AI. Research on hierarchical planning goes back decades (Dean, Firby, and Miller 1988; Erol, Hendler, and Nau 1994; 1996), but is still relevant nowadays (Alford et al. 2016; Xiao et al. 2017). A hierarchical plan is a method for representing problems of automated planning in which the dependency among tasks can be given in the form of networks.

In this work we focus on hierarchical plans represented by task trees, in which the leaves are *primitive* actions (or tasks), and the internal nodes are either *sequence* or *parallel*. The plans we deal with are of stochastic nature, where the duration of a primitive action is given by a random variable. A sequence node denotes a series of tasks that should be performed consecutively, whereas a parallel node denotes a set of tasks that begin at the same time. A *valid* plan is one that is fulfilled before some given *deadline*, i.e., its *makespan* is less than or equal to the deadline. The objective in this context is to compute the probability that a given plan is valid, or more formally computing $P(X < T)$, where X is a random variable representing the makespan of the plan and T is the deadline. We note that deadlines correspond to the resource of *time*, where in fact this work may be applied (possibly with some adjustments) to other resources like fuel, cost, and memory consumption.

As said above, resource consumption (task duration) is uncertain, and described as probability distributions in the leaf nodes. We assume that the distributions are independent but *not* necessarily identically distributed and that the random variables are discrete and have a finite support.

The problem of finding the probability that a task tree satisfies a deadline is known to be NP-hard. In fact, even the problem of summing a set of random variables is NP-hard (Möhring 2001). This is an example of an explicitly given random variable that we need to estimate deadline meeting probabilities for.

Various hierarchical plans that can be represented as task trees with deadlines can be found in the JSHOP2 planner (Nau et al. 2003). One on the examples there is the domain of logistic problems consists of packages that are to be transported by trucks or airplanes. Figure 1 presents an example, which includes one parallel node (packages delivered in parallel), with all descendant task nodes being sequential plans. Given a specific task tree (generated by JSHOP2 or some other planner) and a deadline, our objective is to compute in reasonable time the probability that the plan represented by the task tree meets the deadline. We used these plans for the experiments described in Section 4 below.

3 An Optimal Approximation of a Random Variable

This section starts with a formal statement of the main mathematical problem, and continues with the $\mathrm{OptApprox}$ algorithm that solves the problem in polynomial time.

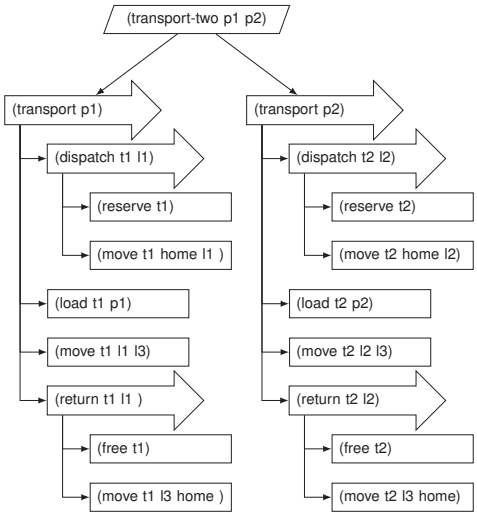

Figure 1: A plan generated by the JSHOP2 algorithm. Arrow shapes represent sequence nodes, parallelograms represent parallel nodes, and rectangles represent primitive nodes.

Problem Statement

We begin with two definitions and then use them to formally state the main algorithmic problem solved in this paper. We use the well known term "discrete random variable" to relate to the outcomes (and the probabilities thereof) of a random phenomenon that has a finite set of possible real values.

Definition 1. *For two discrete random variables X_1 and X_2, we say that X_2 is a* one-sided Kolmogorov approximation *of X_1 with the parameters ε and m, denoted by $X_1 \preceq_{\varepsilon,m} X_2$, if $0 \leq F_{X_2}(t) - F_{X_1}(t) \leq \varepsilon$, for all t, and if $|\operatorname{support}(X_2)| \leq m$.*

Definition 2. *For a random variable X and $m \in \mathbb{N}$, let $\varepsilon^* = \varepsilon^*(X,m) = \inf\{\varepsilon \colon \exists X'.X \preceq_{\varepsilon,m} X'\}$ be the* optimal approximation error *for X with a random variable whose support is of size m.*

The algorithmic problem we focus on is: Given a random variable X with a finite support and a natural number m, find a random variable X' such that $X \preceq_{\varepsilon^*,m} X'$.

The OptApprox Algorithm

In this sub-section we describe an algorithm that solves the problem stated above in polynomial time and memory complexities. The main trick in our method is the observation that we can focus the search for an optimal approximation of a variable X on a small set of candidate variables that we call consecutive approximations of X, defined as follows.

Definition 3. *A partition $P = \{B_1, \ldots, B_n\}$ of a set $S \subseteq \mathbb{R}$ is called* consecutive *if $B_i = [\min B_i, \max B_i] \cap S$ for all i.*

Definition 4. *We say that a random variable X' is a* consecutive approximation *of a random variable X if there is a consecutive partition $P = \{B_1, \ldots, B_n\}$ of $\operatorname{support}(X)$ such that the probability mass function (PMF) of X' is*

$$f_{X'}(t) = \begin{cases} Pr(X \in B_i) & \text{if } t = \min(B_i) \text{ for some } i; \\ 0 & \text{otherwise.} \end{cases}$$

In terms of CDFs, for a given t, we identify the B_i, $1 \leq i < n$, if exists, such that $t \in [\min(B_i), \min(B_{i+1}))$ and define $F_{X'}(t) = F_X(\max(B_i))$. If $t < \min(B_1)$ or $t \geq \min(B_n)$, we define $F_{X'}(t) = 0$ and $F_{X'}(t) = 1$ respectively.

Example 1. *To demonstrate the above notions, we list two possible consecutive approximations of the random variable X defined by its PMF, as follows.*

$$f_X(t) = \begin{cases} 1/3 & \text{if } t = 1 \text{ or } t = 2; \\ 1/6 & \text{if } t = 3 \text{ or } t = 4; \\ 0 & \text{otherwise.} \end{cases}$$

We have $\operatorname{support}(X) = \{1,2,3,4\}$. *One consecutive partition of it is $P = \{\{1,2\},\{3,4\}\}$. The corresponding consecutive approximation is the random variable X_P whose PMF is:*

$$f_{X_P}(t) = \begin{cases} 2/3 & \text{if } t = 1; \\ 1/3 & \text{if } t = 3; \\ 0 & \text{otherwise.} \end{cases}$$

One may also define a different consecutive partition $P' = \{\{1,2,3\},\{4\}\}$ that gives $X_{P'}$ whose PMF is:

$$f_{X_{P'}}(t) = \begin{cases} 5/6 & \text{if } t = 1; \\ 1/6 & \text{if } t = 4; \\ 0 & \text{otherwise.} \end{cases}$$

Alternatively, one can use the CDF notation for $F_{X_P}(t)$ and $F_{X'_P}(t)$ as follows:

$$F_{X_P}(t) = \begin{cases} 0 & \text{if } t < 1; \\ 2/3 & \text{if } 1 \leq t < 3; \\ 1 & \text{if } t \geq 3. \end{cases} \qquad F_{X_{P'}}(t) = \begin{cases} 0 & \text{if } t < 1; \\ 5/6 & \text{if } 1 \leq t < 4; \\ 1 & \text{if } t \geq 4. \end{cases}$$

The next theorem establishes that there is an optimal approximation that is also consecutive. It is key to the correctness proof of the OptApprox algorithm that efficiently scans all possible consecutive approximations to find an optimal one, as we will elaborate later.

Theorem 1. *For any discrete random variable X and any $m \in \mathbb{N}$, there exists a consecutive approximation X_P of X such that $X \preceq_{\varepsilon^*,m} X_P$.*

Proof. Let X' be such that $X \preceq_{\varepsilon^*,m} X'$. Specifically, for all t,

$$F_X(t) \leq F_{X'}(t) \leq F_X(t) + \varepsilon^* \tag{1}$$

The proof is constructive and consists of two steps: (1) we first construct a variable X'' from X' that approximates X as X' does, i.e., $X \preceq_{\varepsilon^*,m} X''$, but also has the property that its support is a subset of the support of X; (2) then, from X'', we construct another random variable, X''', that in addition to being an approximation of X with the same parameters is also equal to X_P for some consecutive partition P.

Assume that $t_1, \ldots, t_n$ are all the elements in the support of X in ascending order. Define the random variable X'' by

$$f_{X''}(t) = \begin{cases} Pr(X' \leq t_1) & \text{if } t{=}t_1; \\ Pr(t_{i-1} < X' \leq t_i) & \text{if } t{=}t_i \text{ for some } i{\neq}1; \\ 0 & \text{otherwise.} \end{cases}$$

In words, given X and X', we construct X'' by defining $f_{X''}(t_i) = Pr(t_{i-1} < X' \leq t_i)$ for all $i = 1, \ldots, n$ (using $t_0 = -\infty$) and $f_{X''}(t) = 0$ for all other ts. Note that $Pr(t_{i-1} < X' \leq t_i)$ may be zero in which case $t_i \notin \text{support}(X'')$.

We will show now that: (1) $\text{support}(X'') \subseteq \text{support}(X)$; (2) $X \preceq_{\varepsilon^*,m} X''$. Since we only assign a non-zero probability to $f_{X''}(t)$ if $t = t_1$ or if $t = t_i$ for some i, i.e., only if t is in the support of X, we have that $\text{support}(X'') \subseteq \text{support}(X)$. Furthermore, if $t_i \in \text{support}(X'')$ then $Pr(t_{i-1} < X' \leq t_i) \neq 0$ which means that there is some $t_{i-1} < t' \leq t_i$ such that $t' \in \text{support}(X')$. To also handle the case where $i = 0$, we denote $t_{-1} = -\infty$. This (unique) mapping gives us that $|\text{support}(X'')| \leq |\text{support}(X')| \leq m$. To complete the proof of the properties of X'', we will show now that $F_X(t) \leq F_{X''}(t) \leq F_{X'}(t)$ for all t by examining the different ts as follows:

(1.1) Case $t < t_1$**:** $F_{X''}(t) = F_X(t) = 0$. Since $F_{X'}(t) \geq 0$ for all t, we get that $F_X(t) \leq F_{X''}(t) \leq F_{X'}(t)$.

(1.2) Case $t = t_i$**:** $F_{X'}(t) = F_{X''}(t)$ and $F_X(t) \leq F_{X'}(t)$ by Eq. (1).

(1.3) Case $t_{i-1} < t < t_i$**:** $F_{X''}(t) = F_{X''}(t_{i-1})$ and $F_X(t) = F_X(t_{i-1})$. Since we already have that $F_X(t_{i-1}) \leq F_{X''}(t_{i-1}) \leq F_{X'}(t_{i-1})$ from Case 1.2, we get that $F_X(t) \leq F_{X''}(t) \leq F_{X'}(t_{i-1})$. By monotonicity of CDF, $F_{X'}(t_{i-1}) \leq F_{X'}(t)$ therefore $F_X(t) \leq F_{X''}(t) \leq F_{X'}(t)$.

(1.4) Case $t > t_n$**:** $F_X(t) = F_{X''}(t) = 1$ and, by Eq. (1), since CDFs are always bounded by one, also $F_{X'}(t) = 1$.

From the four different cases of t, as we already established that $|\text{support}(X'')| \leq m$, we get that $X \preceq_{\varepsilon^*,m} X''$.

Therefore,

$$F_X(t) \leq F_{X''}(t) \leq F_{X'}(t) \leq F_X(t) + \varepsilon^* \qquad (2)$$

Let $s_1, \ldots, s_k$ be the elements in the support of X'' in ascending order $k \leq m$. Define the random variable X'''

$$f_{X'''}(t) = \begin{cases} Pr(s_i \leq X < s_{i+1}) & \text{if } t{=}s_i \text{ for some } i{<}k; \\ Pr(X \geq s_k) & \text{if } t{=}s_k; \\ 0 & \text{otherwise.} \end{cases}$$

We will show that: (1) $X \preceq_{\varepsilon^*,m} X'''$; (2) There exists a partition P such that $X''' = X_P$. Again, we will show that $F_X(t) \leq F_{X'''}(t) \leq F_{X''}(t)$ for all t by examining the different values of t as follows:

(2.1) Case $t < s_1$**:** $F_{X'''}(t) = F_{X''}(t) = 0$ and $F_X(t) \leq F_{X''}(t)$ fro Eq. (2).

(2.2) Case $t = s_i$**:** First, $F_X(t) \leq F_{X'''}(t)$ since $F_{X'''}(s_i) = F_X(s_i) + Pr(s_i < X < s_{i+1})$. Second we show that $F_{X'''}(t) \leq F_{X''}(t)$. Since $X \preceq_{\varepsilon^*,m} X''$, $F_X(s_i) + Pr(s_i < X < s_{i+1}) \leq Pr(X'' < s_{i+1})$. As $s_1, \ldots, s_k$ is the support of X'', $Pr(X'' < s_{i+1}) = F_{X''}(s_i)$. By definition $F_{X'''}(s_i) = F_X(s_i) + Pr(s_i < X < s_{i+1})$. Together we get that $F_{X'''}(s_i) \leq F_{X''}(s_i)$. For the case $t = s_k$, the argument holds with the notation $k + 1 = \infty$.

(2.3) Case $s_{i-1} < t < s_i$**:** $F_{X''}(t) = F_{X''}(s_{i-1})$ and $F_{X'''}(t) = F_{X'''}(s_{i-1})$ therefore by Case 2.2, $F_{X'''}(t) \leq F_{X''}(t)$. Also, $F_X(t) \leq Pr(X < s_i) = F_{X'''}(t)$.

(2.4) Case $t > s_k$**:** $F_{X''}(t) = F_{X'''}(t) = 1$. Since CDFs are always smaller or equal to one, also $F_X(t) \leq 1$.

From the different cases of t, because $\text{support}(X'') = \text{support}(X''')$, we established that $X \preceq_{\varepsilon^*,m} X'''$. The next step is to prove that $X''' = X_P$, by presenting a partition P. As shown before, $\text{support}(X) = \{t_1, \ldots, t_n\}$, $\text{support}(X''') = \{s_1, \ldots, s_k\}$, and $\text{support}(X'') \subseteq \text{support}(X)$ since $\text{support}(X'')$ is equal to $\text{support}(X''')$ then $\text{support}(X''') \subseteq \text{support}(X)$. In addition, $\forall 0 \leq i \leq k, Pr(X''' = s_i) = Pr(s_i \leq X \leq s_{i+1})$ therefore we get that $X''' = X_P$ is a consecutive approximation of X with the partition $P = \{[s_i, s_{i+1}) \cap \text{support}(X) \colon i = 1, \ldots, n-1\}$. □

To demonstrate how the above proof works, the following example shows one possible list of the random variables constructed in the different stages of the proof.

Example 2. *Consider the following two random variable X and X' such that $X \preceq_{\varepsilon^*,m} X'$ where $m = 3$:*

$$f_X(t) = \begin{cases} 1/3 & \text{if } t = 1 \text{ or } t = 2; \\ 1/6 & \text{if } t = 3 \text{ or } t = 4; \\ 0 & \text{otherwise.} \end{cases} \qquad f_{X'}(t) = \begin{cases} 1/6 & \text{if } t = 0.9; \\ 1/2 & \text{if } t = 1; \\ 1/3 & \text{if } t = 2; \\ 0 & \text{otherwise.} \end{cases}$$

In this case $\varepsilon^* = 1/3$ and indeed $X \preceq_{\varepsilon^*,m} X'$. The steps in the proof of Theorem 1 transform X' to a partitioned random variable X''' with the same approximation qualities:

$$f_{X''}(t) = \begin{cases} 2/3 & \text{if } t = 1; \\ 1/3 & \text{if } t = 2; \\ 0 & \text{otherwise.} \end{cases} \qquad f_{X'''}(t) = \begin{cases} 2/3 & \text{if } t = 1; \\ 1/3 & \text{if } t = 3; \\ 0 & \text{otherwise.} \end{cases}$$

Note that this example shows that it may be that X'' is not a consecutive approximation of X. We only guarantee that $\text{support}(X'') \subseteq \text{support}(X)$ and that $X \preceq_{\varepsilon^*,m} X''$. Only the variable X''', in addition to these properties, is guaranteed to be an consecutive approximation. In this case, with the partition $P = \{\{1, 2\}\{3, 4\}\}$.

Theorem 1 gives us that there is a consecutive approximation that is also optimal. This observation allows to restricts the focus of the OptApprox algorithm to only search for the best consecutive approximation.

Chakravarty, Orlin, and Rothblum (1982) proposed a polynomial-time method that, given certain objective functions (additive), finds an optimal consecutive partition. Their method involves the construction of a graph such that the (consecutive) set partitioning problem is reduced to the problem of finding the shortest path in that graph.

The OptApprox algorithm (Algorithm 1) starts by constructing a directed weighted graph G similar to the method of Chakravarty, Orlin, and Rothblum (1982). The nodes V consist of the support of X together with an extra node ∞ for technical reasons, whereas the edges E connect every pair of nodes in one direction (lines 1-2). The weight w of each edge $e = (i, j) \in E$ is determined by the probability of X to get a value between i and j, non inclusive

(lines 3-4), i.e., $w(e) = Pr(i < X < j)$. The values taken are non inclusive, since we are interested only in the error value. The source node of the shortest path problem at hand corresponds to the minimal value in $\mathrm{support}(X)$, and the target node is the extra node ∞. The set of all solution paths in G, i.e., those starting at s and ending in ∞ with at most m edges, is called $paths(G)$. The goal is to find the path in $paths(G)$ with the lightest bottleneck (lines 5-6). This can be achieved by using the $Bellman - Ford$ algorithm with two tweaks. The first is to iterate the graph G in order to find only paths with length of at most m edges. The second is to find the lightest bottleneck as opposed to the traditional objective of finding the shortest path. This is performed by modifying the manner of "relaxation" to $bottleneck(x) = min[max(bottleneck(v), w(e))]$, done also in (Shufan, Ilani, and Grinshpoun 2011). Consequently, we find the lightest maximal edge in a path of length $\leq m$, which represents the minimal error, ε^*, defined in Definition 2. X_P is then derived from the resulting path (lines 7-8).

Algorithm 1: $\mathrm{OptApprox}(X, m)$

1 $S = \mathrm{support}(X) \cup \{\infty\}$
2 $G = (V, E) = (S, \{(i, j) \in S^2 \colon j > i\})$
3 **foreach** $e = (i, j) \in E$ **do**
4 $\quad w(e) = Pr(i < X < j)$
5 /* The following can be obtained, e.g., using the Bellman-Ford algorithm */
6 $l^* = \mathrm{argmin}_{l \in paths(G), |l| \leq m} \max\{w(e) \colon e \in l\}$
7 **foreach** $e = (i, j) \in l^*$ **do**
8 $\quad f_{X'}(i) = Pr(i \leq X < j)$
9 **return** X'

Now we proceed to prove the correctness (optimality) of the $\mathrm{OptApprox}$ algorithm.

Theorem 2. $X \preceq_{\varepsilon^*, m} \mathrm{OptApprox}(X, m)$.

Proof. According to Theorem 1 there exists a consecutive partition P for which $X \preceq_{\varepsilon^*, m} X_P$. For every consecutive partition P there is a path l, $l \in paths(G), |l| \leq m$, such that its corresponding partitioned random variable X_P satisfies $X \preceq_{\varepsilon, m} X_P$ where $\varepsilon = \max\{w(e) \colon e \in l\}$. By using, for instance, the Bellman-Ford algorithm (lines 5-6), we obtain the optimal path l^* containing the minimal edge among all maximal edges of all the paths in $paths(G)$. The optimal consecutive partition P^* associated with this "lightest" path l^*, results in $X' = X_{P^*}$ (lines 7-8). Thus, $X' = \mathrm{OptApprox}(X, m)$ and $X \preceq_{\varepsilon^*, m} X'$. □

Next we analyze the run-time and memory complexity of the $\mathrm{OptApprox}$ algorithm. Note that G is acyclic, therefore a shortest path can be found in $O(E + V)$ time using topological sorting (Christofides 1975).

Theorem 3. *The* $\mathrm{OptApprox}(X, m)$ *algorithm runs in time* $O(n^3)$*, using* $O(n^2)$ *memory where* $n = |\mathrm{support}(X)|$.

Proof. Constructing the graph G takes $O(n^3)$. The number of edges is $O(E) \approx O(n^2)$ and for every edge the weight is at most the sum of all probabilities between the source node s and the target node ∞, $Pr(s < X < \infty)$, $O(n^3)$. The construction is also the only stage that requires memory allocation, specifically $O(E + V) = O(n^2)$. Finding the shortest path takes $O(m(E + V)) \approx O(mn^2)$. Since G acyclic finding shortest path takes $O(E + V)$. We only need to find paths of length $\leq m$, which takes $O(m(E + V))$. Deriving the new random variable X' from the computed path l^* takes $O(mn)$. For every node in l^* (at most m nodes), calculating the probability $P(s < X < \infty)$ takes at most n. To conclude, the worst case run-time complexity is $O(n^3 + mn^2 + mn) = O(n^3)$ and memory complexity is $O(E + V) = O(n^2)$. □

4 Empirical Evaluation

In the previous section, we proved that $\mathrm{OptApprox}$ gives an optimal approximation, i.e., a random variable with minimal one-sided Kolmogorov error. Although locally optimal, it is of interest to analyze the long-term effect of using the $\mathrm{OptApprox}$ algorithm within the process of solving complex problems. Thus, in our main experiment, we use the $\mathrm{OptApprox}$ algorithm as an operator within the wider approximation algorithm proposed in (Cohen, Shimony, and Weiss 2015) for analyzing deadline probabilities in hierarchical plans (see Section 2). It is important to note that while $\mathrm{OptApprox}$ is optimal when applied on a single random variable, the overall algorithm that uses it is not necessarily optimal as an approximation of the completion time of a task tree. Therefore, we aim at empirically evaluating the performance of the $\mathrm{OptApprox}$ algorithm and compare it to other alternative approaches. In an additional experiment we focus on the advantage of using the $\mathrm{OptApprox}$ algorithm on single random variables, and we conclude our experiments with an examination of the problem sizes in which it becomes beneficial in terms of run-time to use the proposed approximation. All solution methods were implemented on Python and the experiments were executed on a hardware comprised of an Intel i5-6500 CPU @ 3.20GHz processor and 8GB memory.

We continue to present the alternative approaches that are evaluated together with the $\mathrm{OptApprox}$ algorithm, followed by the experiments and their results.

Evaluated Alternative Approaches

Our results compared to the exact computation of the CDF, to a simple stochastic sampling scheme, and to a different one-sided Kolmogorov approximation. The exact computation of the CDF is used as reference in order to compute the one-sided Kolmogorov error, i.e. to evaluate how good is our estimation across all possible deadlines. The stochastic sampling scheme is a repeated simulation in which we sample the distribution of every primitive task and aggregate the duration according to the tree structure. This approach enables to derive a distribution over all of the duration results. The last approach, is an implementation of the algorithm proposed by Cohen, Shimony and Weiss (2015) of a non optimal operator called Trim, which returns a random variable that approximates the input by no more than a prescribed

error, ε (Cohen, Shimony, and Weiss 2015). There are two variants of the Trim operator, an upper and a lower error bounds. Here we present the results of the lower bound variant that provides a one-sided Kolmogorov approximation. Both the Trim and $\mathrm{OptApprox}$ are used as operators for achieving one-sided Kolmogorov approximation within the approximation algorithm. The following example illustrates the difference between the Trim and $\mathrm{OptApprox}$ operators.

Example 3. *Consider the variable X and its approximation, X_{Trim}, using* Trim *with $\varepsilon = 1/3$:*

$$f_X(t) = \begin{cases} 0.1 & \text{if } t \in \{1,2,3,4\}; \\ 0.2 & \text{if } t = 5; \\ 0.4 & \text{if } t = 6; \\ 0 & \text{otherwise.} \end{cases} \qquad f_{X_{\mathrm{Trim}}}(t) = \begin{cases} 0.4 & \text{if } t = 1; \\ 0.2 & \text{if } t = 5; \\ 0.4 & \text{if } t = 6; \\ 0 & \text{otherwise.} \end{cases}$$

In this case we get that $d_K(X_{\mathrm{Trim}} - X) = 0.3$. To improve this, using OptApprox *with $m = 3$, we construct $X_{\mathrm{OptApprox}}$ such that $d_K(X_{\mathrm{OptApprox}} - X) = 0.2$:*

$$f_{X_{\mathrm{OptApprox}}}(t) = \begin{cases} 0.3 & \text{if } t = 1 \text{ or } t = 4; \\ 0.4 & \text{if } t = 6; \\ 0 & \text{otherwise.} \end{cases}$$

Although both operators return a one-sided Kolmogorov approximation for a given random variable, they set different parameters in the process. The Trim operator sets the error bound ε and provides guarantees regarding the support size, whereas $\mathrm{OptApprox}$ sets the support size and provides the minimal error. In order to compare the two operators we show a connection between the error bound ε and the support size m. Recall the notation $\varepsilon^*(X, m)$, given in Definition 2, for the quality of the best possible approximation of X with a random variable with support of size m.

Lemma 4. $\varepsilon^*(X, m) \leq 1/m$.

Proof. According to Lemma 2 in (Cohen, Shimony, and Weiss 2015) $X' = \mathrm{Trim}(X, 1/m)$. Lemma 3 in that paper states that for a given error ε the support size m is bounded by $1/\varepsilon$, thus $\varepsilon \leq 1/m$. Turning to the terminology of the present work, this means that $X \preceq_{1/m,m} X'$. Since $\varepsilon^*(X, m)$ is the minimal distance between any random variable with support size m and the random variable X, we get $\varepsilon^*(X, m) \leq 1/m$. □

Cohen, Shimony and Weiss (2015) use a slightly different notation for the error between X and X', $X \preceq_\varepsilon X'$, relating only to the approximation error, without the support size. The following theorem relates $\mathrm{OptApprox}$ to this notation:

Theorem 5. *If $X' = \mathrm{OptApprox}(X, 1/\varepsilon)$ then $X \preceq_\varepsilon X'$.*

Proof. Following the optimality of $\mathrm{OptApprox}$ (Theorem 2), $X \preceq_{\varepsilon^*(X,1/\varepsilon),1/\varepsilon} X'$. Now, given Lemma 4, $X \preceq_{\varepsilon,1/\varepsilon} X'$, and consequently $X \preceq_\varepsilon X'$. □

The above properties enable a fair and sound comparison between the $\mathrm{OptApprox}$ and Trim operators. Following Lemma 4 we set the support size given as the input of $\mathrm{OptApprox}$ to $m = 1/\varepsilon$, whereas ε is the input of Trim. From Theorem 5 we establish that by setting the support size to be $1/\varepsilon$ we bound the error of $\mathrm{OptApprox}$ to at most ε, resulting in the same error guarantees (bounds) by both operators. It remains to determine the actual error resulting from each operator.

Experiments and Results

In the first experiment we focus on the problem of task trees with deadlines, and consider three types of task trees. The first type includes logistic problems of transporting packages by trucks and airplanes (from IPC2 http://ipc.icaps-conference.org/). Hierarchical plans of those logistic problems were generated by the JSHOP2 planner (Nau et al. 2003) (see example problem, Figure 1). The second type consists of task trees used as execution plans for the ROBIL team entry in the DARPA robotics challenge (DRC simulation phase), and the third type is of linear plans (sequential task trees). The primitive tasks in all the trees are modeled as discrete random variables with support of size M obtained by discretization of uniform distributions over various intervals. The number of tasks in a tree is denoted by N.

We fed the approximation algorithm for solving the deadline problem with two different methods of the one-sided Kolmogorov approximation – the $\mathrm{OptApprox}$ and the Trim operators. The parameter m of $\mathrm{OptApprox}$ corresponds to the inverse of ε given to the Trim operator. Note that in order to obtain some error ε, one must take into consideration the size of the task tree, N, therefore, $m/N = 1/(\varepsilon \cdot N)$ (see Lemma 4 and Theorem 5). We ran the exact algorithm as reference, the approximation algorithm using $\mathrm{OptApprox}$ as its operator with $m = 10 \cdot N$, the approximation algorithm using the Trim operator with $\varepsilon = 0.1/N$, and two simple simulations, with a different samples number $s = 10^4$ and $s = 10^6$.

Task Tree	M	OptApprox	Trim	Sampling	
		m/N=10	$\varepsilon \cdot N$=0.1	s=10^4	s=10^6
Logistics (N=34)	2	0	0.0019	0.007	0.0009
	4	0.0046	0.0068	0.0057	0.0005
Logistics (N=45)	2	0.0005	0.002	0.015	0.001
	4	0.003	0.004	0.008	0.0006
DRC-Drive (N=47)	2	0.004	0.009	0.0072	0.0009
	4	0.008	0.019	0.0075	0.0011
Sequential (N=10)	4	0.024	0.04	0.008	0.0016
	10	0.028	0.06	0.0117	0.001

Table 1: Comparison of estimation errors with respect to the reference exact computation on various task trees.

Table 1 shows the results of the main experiment. The quality of the solutions provided by using the $\mathrm{OptApprox}$ operator are better (lower errors) than those provided by the Trim operator, following the optimality guarantees, but is interesting to see that the quality gaps happen in practice in each of the examined task trees. However, in some of the task trees the sampling method produced better results than the approximation algorithm with $\mathrm{OptApprox}$. Nevertheless, the approximation algorithm comes with an inherent advantage of providing an exact quality guarantees, as opposed to the probabilistic guarantees provided by sampling.

m	OptApprox	Trim	Relative error
2	0.491	0.493	0.4%
4	0.242	0.247	2.1%
8	0.118	0.123	4.4%
10	0.093	0.099	6%
20	0.043	0.049	15%
50	0.013	0.019	45.4%

Table 2: OptApprox vs. Trim on randomly generated random variables with original support size $M = 100$.

m	OptApprox	Trim	Relative error
50	0.0193	0.0199	3.4%
100	0.0093	0.0099	7.1%
200	0.0043	0.0049	15.7%

Table 3: OptApprox vs. Trim on randomly generated random variables with original support size $M = 1000$.

In order to better understand the quality gaps in practice between OptApprox and Trim, we investigate their relative errors when applied on single random variables with different sizes of the support (M), and different support sizes of the resulting random variable approximation (m). In each instance of this experiment, a random variable is randomly generated by choosing the probabilities of each element in the support from a uniform distribution and then normalizing these probabilities so that they sum to one.

Tables 2 and 3 present the error produced by OptApprox and Trim on random variables with supports sizes of $M = 100$ and $M = 1000$, respectively. The depicted results in these tables are averages over several instances of random variables for each entry (50 instances in Table 2 and 10 instances in Table 3). The two central columns in each table show the average error of each method, whereas the right column presents the average percentage of the relative error of the Trim operator with respect to the error of the optimal approximation provided by OptApprox; the relative error of each instance is calculated by $(\text{Trim} \,/\, \text{OptApprox}) - 1$. According to the depicted results it is evident that increasing the support size of the approximation m reduces the error, as expected, in both methods. However, the interesting phenomenon is that the relative error percentage of Trim grows with the increase of m.

The above experiments display the quality of approximation provided by the OptApprox algorithm, but it comes with a price tag in the form of run-time performance. The time complexity of both the Trim operator and the sampling method is linear in the number of variables, resulting in much faster run-time performances than OptApprox, for which the time complexity is only polynomial (Theorem 3), not linear. The run-time of the exact computation, however, may grow exponentially. Therefore, we examine in the next experiment the problem sizes in which it becomes beneficial in terms of run-time to use the proposed approximation.

Figure 2 presents a comparison of the run-time performances of an exact computation and approximated computations with OptApprox and Trim as operators. The computation is a summation of a sequence of random variables with support size of M=10, where the number N of variables varies from 6 to 19. In this experiment, we executed the OptApprox operator with m=10 after performing each convolution between two random variables, in order to maintain a support size of 10 in all intermediate computations. Equivalently, we executed the Trim operator with $\varepsilon = 0.1$. The results clearly show the exponential run-time of the exact computation, caused by the convolution between two consecutive random variables. In fact, in the experiment with N=20, the exact computation ran out of memory. These results illuminate the advantage of the proposed OptApprox algorithm that balances between solution quality and run-time performance – while there exist other, faster, methods (e.g., Trim), OptApprox provides high-quality solutions in reasonable (polynomial) time, which is especially important when an exact computation is not feasible, due to time or memory.

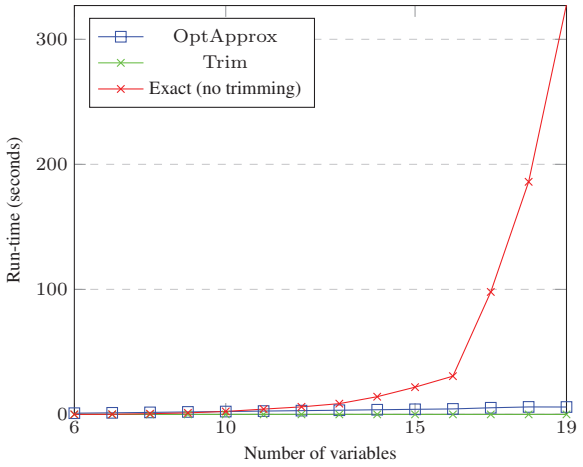

Figure 2: Run-time of a long computation with OptApprox, with Trim, and without any trimming (exact computation).

5 Discussion and Related Work

Compact representations of distributions is mentioned in the literature in various contexts. The most common is for representing continuous random variables with discrete approximations, as in the, so called, three point approximations (Keefer and Bodily 1983; Keefer 1994). Other proposed approximation approaches include the approach presented by Miller III and Rice (1983) and by Hammond and Bickel (2013), where the support of a distribution is partitioned and the mean or the median of every partition is chosen to be in the support of the discrete approximation. Another approach is to compute a discrete variable that has the same moments as the original distribution (Miller III and Rice 1983; Smith 1993). An approach that is applicable when the input distribution is not specified completely involves resolving ambiguities in the definition of the discrete approximation of size m using maximization of the entropy (Rosenblueth, Karmeshu, and Hong 1987).

The contribution of the present paper to this literature is an approximation scheme similar to those proposed

in (Miller III and Rice 1983) and in (Hammond and Bickel 2013). The similarity is in the fact that our proposal is also based on a (consecutive) partition of the support of the input variable (discrete in our case). The difference is that we are proposing to take the minimum of each partition as a representative to be included in the support of the approximation. As with the moments preserving approximations (Miller III and Rice 1983; Smith 1993), where the motivation is that certain objective functions depend mostly on the first moments, our motivation is to discretize such that probabilities of meeting deadlines are preserved. We showed in this paper that a partition that yields an optimal approximation in this sense can be found in polynomial time. As elaborated in the paper, our algorithm improves on the approach of Cohen, Shimony and Weiss (2015) in that it finds an optimal approximation, but it runs in polynomial time where their algorithm runs in linear time.

We view this paper as a first step in the examination of algorithms for optimal approximations of random variables. Beyond the one-sided Kolmogorov measure studied here for the purpose of estimating deadline meeting probabilities, we believe that similar approaches may apply also to the Kolmogorogorv distance, to the Wasserstein distance, and to other measures of approximations for other purposes.

Acknowledgments. This research was supported by the Israeli Ministry of Science and Technology, by the Lynn and William Frankel Center for Computer Science at Ben-Gurion University, and by the Israel Science Foundation (ISF), grants no. 856/16 and 857/12. Liat Cohen is also affiliated with Ariel University under grant no. 3-12802 of the Ministry of Science and Technology.

References

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016. Hierarchical planning: Relating task and goal decomposition with task sharing. In *IJCAI*, 3022–3029.

Beck, J. C., and Wilson, N. 2007. Proactive Algorithms for Job Shop Scheduling with Probabilistic Durations. *Journal of Artificial Intelligence Research* 28:183–232.

Buyya, R.; Garg, S. K.; and Calheiros, R. N. 2011. SLA-oriented resource provisioning for cloud computing: Challenges, architecture, and solutions. In *International Conference on Cloud and Service Computing (CSC)*, 1–10.

Chakravarty, A.; Orlin, J.; and Rothblum, U. 1982. A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations Research* 30(5):1018–1022.

Christofides, N. 1975. *Graph theory: An algorithmic approach*. Academic Press.

Cohen, L.; Shimony, S. E.; and Weiss, G. 2015. Estimating the probability of meeting a deadline in hierarchical plans. In *IJCAI*, 1551–1557.

Dean, T.; Firby, R. J.; and Miller, D. 1988. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence* 4(3):381–398.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, 1123–1128.

Erol, K.; Hendler, J.; and Nau, D. S. 1996. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence* 18(1):69–93.

Fu, N.; Varakantham, P.; and Lau, H. C. 2010. Towards Finding Robust Execution Strategies for RCPSP/max with Durational Uncertainty. In *ICAPS*, 73–80.

Hagstrom, J. N. 1988. Computational complexity of PERT problems. *Networks* 18(2):139–147.

Hammond, R. K., and Bickel, J. E. 2013. Reexamining discrete approximations to continuous distributions. *Decision Analysis* 10(1):6–25.

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

Keefer, D. L., and Bodily, S. E. 1983. Three-point approximations for continuous random variables. *Management Science* 29(5):595–609.

Keefer, D. L. 1994. Certainty equivalents for three-point discrete-distribution approximations. *Management Science* 40(6):760–773.

Lilliefors, H. W. 1967. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association* 62(318):399–402.

Miller III, A. C., and Rice, T. R. 1983. Discrete approximations of probability distributions. *Management Science* 29(3):352–362.

Möhring, R. 2001. Scheduling under uncertainty: Bounding the makespan distribution. *Computational Discrete Mathematics* 79–97.

Nau, D. S.; Au, T.-C.; Ilghami, O.; Kuter, U.; Murdock, J. W.; Wu, D.; and Yaman, F. 2003. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research* 20:379–404.

Pavlikov, K., and Uryasev, S. 2016. CVaR distance between univariate probability distributions and approximation problems. Technical Report 2015-6, University of Florida.

Rosenblueth, E.; Karmeshu; and Hong, H. P. 1987. Maximum entropy and discretization of probability distributions. *Probabilistic Engineering Mechanics* 2(2):58–63.

Shperberg, S. S.; Shimony, S. E.; and Felner, A. 2017. Monte-Carlo tree search using batch value of perfect information. In *UAI*.

Shufan, E.; Ilani, H.; and Grinshpoun, T. 2011. A two-campus transport problem. In *MISTA*, 173–184.

Smith, J. E. 1993. Moment methods for decision analysis. *Management Science* 39(3):340–358.

Vallender, S. 1974. Calculation of the Wasserstein distance between probability distributions on the line. *Theory of Probability & Its Applications* 18(4):784–786.

Xiao, Z.; Herzig, A.; Perrussel, L.; Wan, H.; and Su, X. 2017. Hierarchical task network planning with task insertion and state constraints. In *IJCAI*, 4463–4469.