
An algorithm for computing an optimal approximation of a random variable with respect to the Kolmogorov distance

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 We present an algorithm that takes a discrete random variable X and a number m
2 and computes a random variable X' whose support (set of possible outcomes) is
3 of size at most m and whose Kolmogorov distance from X is minimal.

4 1 Introduction

5 Many different approaches to approximation of probability distributions are studied in the litera-
6 ture [12, 15, 16]. The approaches vary in the types random variables involved, how they are rep-
7 resented, and in the criteria used for evaluation of the quality of the approximations. This paper is
8 on approximating discrete distributions represented as explicit probability mass functions with ones
9 that are simpler to store and to manipulate. This is needed, for example, when a discrete distribution
10 is given as a large data-set, obtained, e.g., by sampling, and we want to represent it approximately
11 with a small table.

12 The main contribution of this paper is an efficient algorithm for computing the best possible ap-
13 proximation of a given random variable with a random variable whose complexity is not above a
14 prescribed threshold, where the measures of the quality of the approximation and the complexity of
15 the random variable are as specified in the following two paragraphs.

16 We measure the quality of an approximation by the distance between the original variable and the
17 approximate one. Specifically, we use the Kolmogorov distance which is one of the most used in
18 statistical practice and literature. Given two random variables X and X' whose cumulative distribu-
19 tion functions (cdfs) are F_X and $F_{X'}$, respectively, the Kolmogorov distance between X and X' is
20 $d_K(X, X') = \sup_t |F_X(t) - F_{X'}(t)|$ (see, e.g., [9]). We say that X' is a good approximation of X
21 if $d_K(X, X')$ is small.

22 The complexity of a random variable is measured by the size of its support, the number of values
23 that it can take, $|\text{support}(X)| = |\{x: \Pr(X = x) \neq 0\}|$. When distributions are maintained as
24 explicit tables, as done in many implementations of statistical software, the size of the support of
25 a variable is proportional to the amount of memory needed to store it and to the complexity of the

26 computations around it. In summary, the exact notion of optimality of the approximation targeted in
 27 this paper is:

28 **Definition 1.** *A random variable X' is an optimal m -approximation of a random variable X if*
 29 *$|\text{support}(X')| \leq m$ and there is no random variable X'' such that $|\text{support}(X'')| \leq m$ and*
 30 *$d_K(X, X'') < d_K(X, X')$.*

31 The main contribution of the paper is an efficient algorithm that takes X and m as parameters and
 32 constructs an optimal m -approximation of X .

33 The rest of the paper is organized as follows. In Section 2 we describe how our work relates to other
 34 algorithms and problems studied in the literature. In Section 3 we detail the proposed algorithm,
 35 analyze its properties, and prove the main theorem. In Section 4 we demonstrate how the proposed
 36 approach performs on the problem of estimating the probability of hitting deadlines is plans and
 37 compare it to alternatives approximation approaches from the literature. We also demonstrate the
 38 performance of our approximation algorithm on some randomly generated random variables. The
 39 paper is concluded with a discussion in Section 5.

40 2 Related Work

41 The most relevant work related to this paper is the papers by Cohen et. al. [5, 4]. These papers study
 42 approximations of random variables in the context of estimating deadlines. In this context, X' is
 43 defined to be a good approximation of X if $F_{X'}(t) > F_X(t)$ for any t and $\sup_t F_{X'}(t) - F_X(t)$
 44 is small. This is not a distance because it is not symmetric. The motivation given by Cohen et. al.
 45 for using this notion is for cases where overestimation of the probability of missing a deadline is
 46 acceptable but underestimation is not. In Section 4, we consider the same examples examined by
 47 Cohen et. al. and show how the algorithm proposed in this paper performs relative to the algorithms
 48 proposed there when both over- and under- estimations are allowed. As expected, the Kolmogorov
 49 distance between the approximation and the original random variable is smaller by a factor of one
 50 half, on average.

51 Another very relevant work is the theory of Sparse Approximation (aka Sparse Representation) that
 52 deals with sparse solutions for systems of linear equations, as follows.

Given a matrix $D \in \mathbb{R}^{n \times p}$ and a vector $x \in \mathbb{R}^n$, the most studied sparse representation problem is
 finding the sparsest possible representation $\alpha \in \mathbb{R}^p$ satisfying $x = D\alpha$:

$$\min_{\alpha \in \mathbb{R}^p} \|\alpha\|_0 \text{ subject to } x = D\alpha.$$

53 where $\|\alpha\|_0 = |\{i : \alpha_i \neq 0, i = 1, \dots, p\}|$ is the ℓ_0 pseudo-norm, counting the number of non-zero
 54 coordinates of α . This problem is known to be NP-Hard with a reduction to NP-complete subset
 55 selection problems.

In these terms, using also the ℓ_∞ norm that represents the maximal coordinate and the ℓ_1 norm that
 represents the sum of the coordinates, our problem can be phrased as:

$$\min_{\alpha \in [0, \infty)^p} \|x - D\alpha\|_\infty \text{ subject to } \|\alpha\|_0 = m \text{ and } \|\alpha\|_1 = 1.$$

56 where D is the all-ones triangular matrix (the entry at row i and column j is one if $i \leq j$ and zero
 57 otherwise), x is related to X such that the i th coordinate of x is $F_X(x_i)$ where $\text{support}(X) =$
 58 $\{x_1 < x_2 < \dots < x_n\}$ and α is related to X' such that the i th coordinate of α is $f_{X'}(x_i)$. The
 59 functions F_X and $f_{X'}$ represent, respectively, the cumulative distribution function of X and the

60 mass distribution function of X' . This, of course, means that the coordinates of x are assumed to
 61 be positive and monotonically increasing and that the last coordinate of x is assumed to be one. We
 62 demonstrate an application for this specific sparse representation problem and show that it can be
 63 solve in $O(n^2m)$ time and $O(m^2)$ memory.

64 Another related research is binning in statistical inference. Consider, for example, the problem
 65 of credit scoring [21] that deals with separating good applicants from bad applicants where the
 66 Kolmogorov–Smirnov statistic KS is a standard measure. The KS comparison is often preceded
 67 by a procedure called binning where a large table is shrunk to a smaller one by collecting nearby
 68 values together. There are many methods for binning [11, 17, 2, 19]. In this context, our algorithm
 69 can be consider as a new binning strategy that provides optimality guarantees with respect to the
 70 Kolmogorov distance that none of the existing binning technique that we are aware of provides.

71 The present study is also related to the work of Pavlikov and Uryasev [15], where a procedure for
 72 producing a random variable X' that optimally approximates a random variable X is presented.
 73 Their approximation scheme, achieved using convex and linear programming, is designed for a
 74 different notion of distance (called CVaR). The new contribution of the present work in this context
 75 is that our method is direct, not using linear or convex programming, thus allowing tighter analysis
 76 of time and memory complexity. Also, our method is designed for optimizing the Kolmogorov
 77 distance that is more prevalent in applications.

78 3 An Algorithm for Optimal Approximation

79 In the scope of this section, let X be a given random variable with a finite support of size n , and let
 80 $0 < m \leq n$ be a given complexity bound. We first develop notations and collect facts towards an
 81 algorithm for finding an optimal m -approximation of X .

82 The first useful fact is that it is enough to limit our search to approximations X' 's such that
 83 $\text{support}(X') \subseteq \text{support}(X)$:

84 **Lemma 2.** *There is an optimal m -approximation X' of X such that $\text{support}(X') \subseteq \text{support}(X)$.*

85 *Proof.* Let X'' be a random variable whose support is of size smaller or equal to m . We find a
 86 random variable X' with $\text{support}(X') \subseteq \text{support}(X)$ such that $d_K(X, X') \leq d_K(X, X'')$. Let
 87 $\{x_1, \dots, x_n\} = \text{support}(X)$, and let $x_0 = -\infty, x_{n+1} = \infty$. Consider the random variable X'
 88 whose probability mass function is $f_{X'}(x_i) = P(x_{i-1} < X'' \leq x_i)$ for $i = 1, \dots, n-1$, $f_{X'}(x_n) =$
 89 $P(x_n - 1 < X'' < x_{n+1})$, and $F_{X'}(x) = 0$ if $x \notin \text{support}(X)$. Since X' "accumulates" the non-
 90 zero probabilities of X'' to the support of X , we have that f is a probability mass function and
 91 therefore X' is well defined.

92 First see by construction that $|F_X(x_i) - F_{X'}(x_i)| = |F_X(x_i) - F_{X''}(x_i)|$ for every $1 \leq i \leq n-1$.
 93 For $i = n$ we have $|F_X(x_n) - F_{X'}(x_n)| = |1 - 1| = 0$. Finally see that $|F_X(x) - F_{X'}(x)| =$
 94 $|F_X(x_i) - F_{X'}(x_i)|$ for every $0 \leq i < n+1$ and $x_i < x < x_{i+1}$. Therefore we have that
 95 $d_K(X, X') = \max_i |F_X(x_i) - F_{X'}(x_i)| \leq \max_i |F_X(x_i) - F_{X''}(x_i)| \leq d_K(X, X'')$. \square

96 Next, note that every random variable X'' with support of size at most m that is contained in
 97 $\text{support}(X)$ can be described by first setting the (at most m) elements of the support of X'' ; then
 98 for every such option, determine X'' by setting probability values for the elements in the chosen
 99 support of X' , and setting 0 for rest of the elements.

100 Denote the set of random variables with support $S \subseteq \text{support}(X)$ by \mathbb{X}_S . In Step 1 below, we find
 101 a random variable in \mathbb{X}_S that minimizes the Kolmogorov distance from X , and denote this distance
 102 by $\varepsilon(X, S)$. Next, in Step 2, that we will describe later, we will show how to efficiently find S
 103 that minimizes $\varepsilon(X, S)$ among all the sets that satisfy $S \subset \text{support}(X)$ and $|S| \leq m$. Then the
 104 minimized random variable \mathbb{X}_S from the minimal S , is the m -optimal approximation to X .

105 3.1 Step 1: Finding an X' in \mathbb{X}_S that minimizes $d_K(X, X')$

106 We first fix a set $S \subseteq \text{support}(X)$ of size at most m , and among all the random variables in
 107 \mathbb{X}_S find one with a minimal distance from X . Denote the elements of S in increasing order by
 108 $S = \{x_1 < \dots < x_m\}$ and let $x_0 = -\infty$, and $x_{m+1} = \infty$. For every $1 \leq i \leq m$ let \hat{x}_i be the
 109 maximal element of $\text{support}(X)$ that is smaller than x_i .

110 Next, as the elements of S are also elements of $\text{support}(X)$, we can define the following weight
 111 function:

112 **Definition 3.** For $0 \leq i \leq m$ let

$$w(x_i, x_{i+1}) = \begin{cases} P(x_i < X < x_{i+1}) & \text{if } i = 0 \text{ or } i = m; \\ P(x_i < X < x_{i+1})/2 & \text{otherwise.} \end{cases}$$

113 Note that $x_i = -\infty$ for $i = 0$ and $x_i = \infty$ for $i = m + 1$. Also note that $P(x_i < X < x_{i+1}) =$
 114 $F_X(\hat{x}_{i+1}) - F_X(x_i)$, a fact that we will use throughout this section.

115 **Definition 4.** Let $\varepsilon(X, S) = \max_{i=0, \dots, m} w(x_i, x_{i+1})$.

116 We first show that $\varepsilon(X, S)$ is a lower bound. That is, every random variable in \mathbb{X}_S has a distance at
 117 least $\varepsilon(X, S)$. Then, we present a random variable $X' \in \mathbb{X}_S$ with distance $\varepsilon(X, S)$. It then follows
 118 that such X' is an optimal m -approximation random variable among all random variables in \mathbb{X}_S .

119 The intuition behind choosing these specific weights and $\varepsilon(X, S)$ being a lower bound is as follows.
 120 Since for every $X' \in \mathbb{X}_S$ the probability values of X' for the elements not in S are set to 0, we have
 121 that $F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i)$. Therefore the distance between X' and X at points x_i and \hat{x}_{i+1} that we
 122 have to take into additional account is increased by $F_X(\hat{x}_{i+1}) - F_X(x_i) = P(x_i < X < x_{i+1})$.

123 Formally we have the following.

124 **Proposition 5.** If $X' \in \mathbb{X}_S$ then $d_K(X, X') \geq \varepsilon(X, S)$.

125 *Proof.* By definition, for every $0 \leq i \leq m$, $d_K(X, X') \geq \max\{|F_X(\hat{x}_{i+1}) -$
 126 $F_{X'}(\hat{x}_{i+1})|, |F_X(x_i) - F_{X'}(x_i)|\}$. Note that $F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i)$ since the probability values
 127 for all the elements not in S are set to 0.

128 If $i = 0$, that is $x_i = -\infty$, we have that $F_X(x_i) = F_{X'}(x_i) = F_{X'}(\hat{x}_{i+1}) = 0$ and therefore
 129 $d_K(X, X') \geq |F_X(\hat{x}_{i+1})| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

130 If $i = m$, that is $x_{i+1} = \infty$, we have that $F_X(\hat{x}_{i+1}) = F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i) = 1$. and therefore
 131 $d_K(X, X') \geq |1 - F_X(\hat{x}_i)| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

132 Otherwise for every $1 \leq i < m$, we use the fact that $\max\{|a|, |b|\} \geq |a - b|/2$ for every $a, b \in$
 133 \mathbb{R} , to have $d_K(X, X') \geq 1/2|F_X(\hat{x}_{i+1}) - F_X(x_i) + F_{X'}(x_i) - F_{X'}(\hat{x}_{i+1})|$. So $d_K(X, X') \geq$
 134 $1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1})/2 = w(x_i, x_{i+1})$.

135 Therefore since $d_K(X, X') \geq w(x_i, x_{i+1})$ for every $0 \leq i \leq m$, by definition of $\varepsilon(X, S)$ proof
 136 follows. \square

Next we show a random variable $X' \in \mathbb{X}_S$ with a distance of $\varepsilon(X, S)$ from X . Thus X' is an optimal m -approximation among the set \mathbb{X}_S . We define X' as follows:

Definition 6. Let $f_{X'}(x_i) = w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)$ for $i = 1, \dots, m$ and $f_{X'}(x) = 0$ for $x \notin S$.

We first show that X' is a properly defined random variable:

Lemma 7. $f_{X'}$ is a probability mass function.

Proof. From definition $f_{X'}(x_i) \geq 0$ for every i . To see that $\sum_i f_{X'}(x_i) = 1$, we have
 $\sum_i f_{X'}(x_i) = \sum_i (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = \sum_{x_i \in S} f_X(x_i) + w(x_0, x_1) +$
 $\sum_{0 < i < m} 2w(x_i, x_{i+1}) + w(x_m, x_{m+1}) = \sum_{x_i \in S} P(X = x_i) + P(x_0 < X < X_1) +$
 $\sum_{0 < i < m} P(x_i < X < X_{i+1}) + P(x_m < X < X_{m+1}) = 1$ since this sum is the entire cdf of X . \square

Note that X' can be constructed in linear time to the size of the cdf of X . Intuitively the setting of X' allows to take an "advantage" of distance from X at the elements of $\text{support}(X')$, to avoid the overall increased distance of X from X' at the elements that are not at $\text{support}(X)$ and in which $f_{X'}$ is set to 0. Formally we have the following.

Lemma 8. Let $x \in \text{support}(X)$ and $0 \leq i \leq m$ be such that $x_i \leq x \leq x_{i+1}$ then $-w(x_i, x_{i+1}) \leq F_X(x) - F_{X'}(x) \leq w(x_i, x_{i+1})$.

Proof. We prove by induction on $0 \leq i < m$.

First see that $F_{X'}(j) = 0$ for every $x_0 < j < x_1$ and therefore $F_X(j) - F_{X'}(j) = F_X(j) - 0 \leq F_X(\hat{x}_1) = F_X(\hat{x}_1) - F_X(x_0) = w(x_0, x_1)$. For $j = x_1$ we have $F_X(x_1) - F_{X'}(x_1) = F_X(\hat{x}_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1)) = w(x_0, x_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1)) = -w(x_1, x_2)$.

Next assume that $F_X(\hat{x}_i) - F_{X'}(\hat{x}_i) = w(x_{i-1}, x_i)$. Then $F_X(x_i) - F_{X'}(x_i) = F_X(\hat{x}_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = w(x_{i-1}, x_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = -w(x_i, x_{i+1})$.

As before we have that for all $x_i < j < x_{i+1}$, we have $F_X(j) - F_{X'}(j) = F_X(j) - F_{X'}(\hat{x}_{i+1}) \leq F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})$. Then $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1}) = (F_X(x_i) + P(x_i < x < x_{i+1})) - F_{X'}(x_i) = -w(x_i, x_{i+1}) + 2w(x_i, x_{i+1}) = w(x_i, x_{i+1})$.

Finally for $x_m \leq j \leq x_{m+1}$ we have that $F_{X'}(x_m) = 1$ therefore $F_X(x_m) - F_{X'}(x_m) = (1 - P(x_m < X < x_{m+1})) - 1 = P(x_m < X < x_{m+1}) = w(x_m, x_{m+1})$, and for every $x_m < j < x_{m+1}$ we have $F_X(j) - F_{X'}(j) < (1 - P(x_m < X < x_{m+1})) - 1 < -P(x_m < X < x_{m+1}) = -w(x_m, x_{m+1})$ as required. \square

From Lemma 8, by the definition of $\varepsilon(X, S)$, we then have:

Corollary 9. $d_K(X, X') = \varepsilon(X, S)$.

3.2 Step 2: Finding an S that minimizes $\varepsilon(X, S)$

Chakravarty, Orlin, and Rothblum [3] proposed a polynomial-time method that, given a 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.

176 The KolmogorovApprox algorithm (Algorithm 1) starts by constructing a directed weighted graph
 177 G similar to the method of Chakravarty, Orlin, and Rothblum [3]. The nodes V consist of the support
 178 of X together with an extra two nodes, $-\infty$ and ∞ for technical reasons, whereas the edges E
 179 connect every pair of nodes in one direction (lines 1-2). The weight w of each edge $e = (x, y) \in E$
 180 is determined by one of two cases as in Definition 3. The values taken are non inclusive, since
 181 we are interested only in the error value. The source node of the shortest path problem at hand
 182 corresponds to the $-\infty$ node added to G in the construction phase, and the target node is the extra
 183 node ∞ . The set of all solution paths in G , i.e., those starting at $-\infty$ and ending in ∞ with at most
 184 m edges, is called $paths(G, -\infty, \infty)$. The goal is to find the path l in $paths(G, -\infty, \infty)$ with the
 185 lightest bottleneck (line 3). This can be achieved by using the *Bellman – Ford* algorithm with
 186 two tweaks. The first is to iterate the graph G in order to find only paths with length of at most m
 187 edges. The second is to find the lightest bottleneck as opposed to the traditional objective of finding
 188 the shortest path. This is performed by modifying the manner of “relaxation” to $bottleneck(x) =$
 189 $\min[\max(bottleneck(v), w(e))]$, done also in [10, 18]. Consequently, we find the lightest maximal
 190 edge in a path of length $\leq m$, which represents the minimal error, $\varepsilon(X, S)$, defined in Definition 4
 191 where the nodes in path l represent the elements in set S . The approximated random variable X'
 192 is then derived from the resulting path l (lines 4-5). Every node $x \in l$ represent a value in the new
 193 calculated random variable X' , we then iterate the path l to find the probability of the event $f_{X'}(x)$
 194 as described in Definition 6.

Algorithm 1: KolmogorovApprox(X, m)

```

1  $S = \text{support}(X) \cup \{\infty, -\infty\}$ 
2  $G = (V, E) = (S, \{(x, y) : x < y\})$ 
3  $(x_0, \dots, x_{m+1}) = l \in \arg\min_{l \in paths(G, -\infty, \infty), |l| \leq m} \max\{w(e) : e \in l\}$ 
4 for  $0 < i < m + 1$  do
5    $f_{X'}(x_i) = w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)$ 
6 return  $X'$ 

```

195 **Theorem 10.** KolmogorovApprox(X, m) is an m -optimal-approximation of X .

196 *Proof.* If we consider the vertexes $S = l \setminus \{-\infty, \infty\}$ for a path $l \in paths(G, -\infty, \infty)$ we have
 197 that $\max\{w(e) : e \in l\} = \varepsilon(X, S)$. Therefore, line 3 of the algorithm essentially computes a set
 198 $S \in \arg\min_{S \subseteq \text{support}(X), |S| \leq m} \varepsilon(X, S)$. By Corollary 9, the variable X' constructed in lines 4 and
 199 5 satisfies $d_K(X, X') = \varepsilon(X, S)$ and by the minimality of S and by Proposition 5, it is an optimal
 200 approximation. \square

201 **Theorem 11.** The KolmogorovApprox(X, m) algorithm runs in time $O(mn^2)$, using $O(n^2)$ mem-
 202 ory where $n = |\text{support}(X)|$.

203 *Proof.* Constructing the graph G takes $O(n^2)$. The number of edges is $O(E) \approx O(n^2)$ and for every
 204 edge the weight is at most the sum of all probabilities between the source node $-\infty$ and the target
 205 node ∞ , which can be done efficiently by aggregating the weights of already calculated edges. The
 206 construction is also the only stage that requires memory allocation, specifically $O(E + V) = O(n^2)$.
 207 Finding the shortest path takes $O(m(E + V)) \approx O(mn^2)$.

208 [[GW: put a reference to the work of the fellows from the Technion to avoid some of this?]]

209 Since G is DAG (directed acyclic graph) finding a shortest path takes $O(E + V)$. We only need
 210 to find paths of length $\leq m$, which takes $O(m(E + V))$. Deriving the new random variable X'
 211 from the computed path l takes $O(m)$. For every node x_i in l (at most m nodes), use the already

calculated weights to find the probability mass function $f_{X'}(x_i)$. To conclude, the worst case run-time complexity is $O(n^2 + mn^2 + m) = O(mn^2)$ and memory complexity is $O(E + V) = O(n^2)$. \square

4 A case study and experimental results

The case study examined in our experiments is the problem of task trees with deadlines [5, 4]. Hierarchical planning is a well-established field in AI [6, 7, 8], and is still relevant nowadays [1, 20]. 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, here we focus on hierarchical plans represented by task trees. The leaves in a task tree are *primitive* actions (or tasks), and the internal nodes are either *sequence* or *parallel* actions. The plans we deal with are of stochastic nature, and the task duration is 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.

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. 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 [13]. This is an example of an explicitly given random variable that we need to estimate deadline meeting probabilities for.

The first experiment we focus on is 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 [14], one parallel node with all descendant task nodes being in sequence. 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 implemented the approximation algorithm for solving the deadline problem with four different methods of approximation. The first two are for achieving a one-sided Kolmogorov approximation – the OptTrim [4] and the Trim [5] operators, and the third is a simple sampling scheme. We used those methods as a comparison to the Kolmogorov approximation with the suggested KolmogorovApprox algorithm. The parameter m of OptTrim and KolmogorovApprox 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, $m/N = 1/(\varepsilon \cdot N)$. We ran also an exact computation as a reference to the approximated one in order to calculate the error. The experiments conducted with the following operators and their parameters: KolmogorovApprox operator with $m = 10 \cdot N$, the OptTrim operator with $m = 10 \cdot N$, the Trim as operator with $\varepsilon = 0.1/N$, and two simple simulations, with a different samples number $s = 10^4$ and $s = 10^6$.

Table 1 shows the results of the case study experiment. The quality of the solutions provided by using the KolmogorovApprox operator are better than those provided by the Trim and OptTrim operators, following the optimality guarantees, but is interesting to see that the quality gaps hap-

Task Tree	M	KolmogorovApprox	OptTrim	Trim	Sampling	
		$m/N=10$	$m/N=10$	$\varepsilon \cdot N=0.1$	$s=10^4$	$s=10^6$
Logistics ($N=34$)	2	0	0	0.0019	0.007	0.0009
	4	0.0024	0.0046	0.0068	0.0057	0.0005
Logistics ($N=45$)	2	0.0002	0.0005	0.002	0.015	0.001
	4	0	0.003	0.004	0.008	0.0006
DRC-Drive ($N=47$)	2	0.0014	0.004	0.009	0.0072	0.0009
	4	0.001	0.008	0.019	0.0075	0.0011
Sequential ($N=10$)	2	0.0093	0.015	0.024	0.0063	0.0008
	4	0.008	0.024	0.04	0.008	0.0016

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

pen 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 KolmogorovApprox. Nevertheless, the approximation algorithm comes with an inherent advantage of providing an exact quality guarantees, as opposed to the probabilistic guarantees provided by sampling.

In order to better understand the quality gaps in practice between KolmogorovApprox, OptTrim, and Trim, we investigate their relative errors when applied on single random variables with support size $n = 100$, 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 uniformly and then normalizing these probabilities so that they sum to 1.

Figure ?? present the error produced by the above methods. The depicted results are averages over several instances (50 instances) of random variables. The curves in the figure show the average error of OptTrim and Trim operators with comparison to the average error of the optimal approximation provided by KolmogorovApprox as a function of m . According to the depicted results it is evident that increasing the support size of the approximation m reduces the error, as expected, in all three methods. However, errors produced by the KolmogorovApprox are significantly smaller, a half of the error produced by OptTrim and Trim.

We also examined how our algorithm compares to linear programing as described and discussed, for example, in [15]. We ran an experiment to compare the run-time between the KolmogorovApprox algorithm with the run-time of a state-of-art implementation of linear programing. We used the “Minimize” function of Wolfram Mathematica and fed it with the equations $\min_{\alpha \in \mathbb{R}^n} \|x - \alpha\|_\infty$ subject to $\|\alpha\|_0 \leq m$ and $\|\alpha\|_1 = 1$. The run-time comparison results were clear and persuasive, for a random variable with support size $n = 10$ and $m = 5$, the LP algorithm run-time was 850 seconds, where the KolmogorovApprox algorithm run-time was less than a tenth of a second. For $n = 100$ and $m = 5$, the KolmogorovApprox algorithm run-time was 0.14 seconds and the LP algorithm took more than a day. Due to these timing results of the LP algorithm we did not proceed to examine it any further. Since it is not trivial to formally analyze the run-time of the LP algorithm, we conclude by the reported experiment that in this case the LP algorithm might not be as efficient as KolmogorovApprox algorithm whose complexity is proven to be polynomial in Theorem 11.

5 Discussion

References

- [1] R. Alford, V. Shivashankar, M. Roberts, J. Frank, and D. W. Aha. Hierarchical planning: Relating task and goal decomposition with task sharing. In *IJCAI*, pages 3022–3029, 2016.
- [2] C. Bolton et al. *Logistic regression and its application in credit scoring*. PhD thesis, Citeseer, 2010.
- [3] A. Chakravarty, J. Orlin, and U. Rothblum. A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations Research*, 30(5):1018–1022, 1982.
- [4] L. Cohen, T. Grinshpoun, and G. Weiss. Optimal approximation of random variables for estimating the probability of meeting a plan deadline. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
- [5] L. Cohen, S. E. Shimony, and G. Weiss. Estimating the probability of meeting a deadline in hierarchical plans. In *IJCAI*, pages 1551–1557, 2015.
- [6] T. Dean, R. J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4(3):381–398, 1988.
- [7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.
- [8] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93, 1996.
- [9] J. D. Gibbons and S. Chakraborti. Nonparametric statistical inference. In *International encyclopedia of statistical science*, pages 977–979. Springer, 2011.
- [10] R. Guérin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM Transactions on Networking (TON)*, 10(5):613–620, 2002.
- [11] E. Mays. *Handbook of credit scoring*. Global Professional Publishi, 2001.
- [12] A. C. Miller and T. R. Rice. Discrete approximations of probability distributions. *Management Science*, 29(3):352–362, 1983.
- [13] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. *Computational Discrete Mathematics*, pages 79–97, 2001.
- [14] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.
- [15] K. Pavlikov and S. Uryasev. CVaR distance between univariate probability distributions and approximation problems. Technical Report 2015-6, University of Florida, 2016.
- [16] A. N. Pettitt and M. A. Stephens. The kolmogorov-smirnov goodness-of-fit statistic with discrete and grouped data. *Technometrics*, 19(2):205–210, 1977.

- 321 [17] M. Refaat. *Credit Risk Scorecard: Development and Implementation Using SAS*. Lulu. com,
322 2011.
- 323 [18] E. Shufan, H. Ilani, and T. Grinshpoun. A two-campus transport problem. In *MISTA*, pages
324 173–184, 2011.
- 325 [19] N. Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*,
326 volume 3. John Wiley & Sons, 2012.
- 327 [20] Z. Xiao, A. Herzig, L. Perrussel, H. Wan, and X. Su. Hierarchical task network planning with
328 task insertion and state constraints. In *IJCAI*, pages 4463–4469, 2017.
- 329 [21] G. Zeng. A comparison study of computational methods of kolmogorov–smirnov statistic in
330 credit scoring. *Communications in Statistics-Simulation and Computation*, 46(10):7744–7760,
331 2017.