
Kolmogorov Approximation

Anonymous Author(s)

Affiliation

Address

email

1 Introduction

Many different approaches to approximation of probability distributions are studied in the literature [10, 13, 14]. The papers vary in the types random variables involved, how they are represented, and in the criteria used for evaluation of the quality of the approximations. This paper is on approximating discrete distributions represented as explicit probability mass functions with ones that are simpler to store and to manipulate. This is needed, for example, when a discrete distribution is given as a large data-set, obtained, e.g., by sampling, and we want to represent it approximately with a small table.

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

We measure the quality of an approximation by the distance between the original variable and the approximate one. Specifically, we use the Kolmogorov distance which is one of the most used in statistical practice and literature. Given two random variables X and X' whose cumulative distribution functions (cdfs) are F_X and $F_{X'}$, respectively, the Kolmogorov distance between X and X' is $d_K(X, X') = \sup_t |F_X(t) - F_{X'}(t)|$ (see, e.g., [8]). We say that X' is a good approximation of X if $d_K(X, X')$ is small.

The complexity of a random variable is measured by the size of its support, the number of values that it can take, $|\text{support}(X)| = |\{x: \Pr(X = x) \neq 0\}|$. When distributions are maintained as explicit tables, as done in many implementations of statistical software, the size of the support of a variable is proportional to the amount of memory needed to store it and to the complexity of the computations around it. In summary, the exact notion of optimality of the approximation targeted in this paper is:

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

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

The rest of the paper is organized as follows. In Section 2 we describe how our work relates to other algorithms and problems studied in the literature. In Section 3 we detail the proposed algorithm,

analyze its properties, and prove the main theorem. In Section 4 we demonstrate how the proposed approach performs on the problem of estimating the probability of hitting deadlines is plans and compare it to alternatives approximation approaches from the literature. We also demonstrate the performance of our approximation algorithm on some randomly generated random variables. The paper is concluded with a discussion in Section 5.

2 Related Work

The problem studied in this paper is related to the theory of Sparse Approximation (aka Sparse Representation) that 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 $x = D\alpha$:

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

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

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

where D is the all-ones triangular matrix (the entry at row i and column j is one if $i \leq j$ and zero otherwise), x is related to X such that the i th coordinate of x is $F_X(x_i)$ where $\text{support}(X) = \{x_1 < x_2 < \dots < x_n\}$ and α is related to X' such that the i th coordinate of α is $f_{X'}(x_i)$. The functions F_X and $f_{X'}$ represent, respectively, the cumulative distribution function of X and the mass distribution function of X' . This, of course, means that the coordinates of x are assumed to be positive and monotonically increasing and that the last coordinate of x is assumed to be one. We demonstrate an application for this specific sparse representation problem and show that it can be solve in $O(n^2m)$ time and $O(m^2)$ memory.

The present study is also a continuation of the work of Pavlikov and Uryasev [13], 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.

3 An Algorithm for Optimal Approximation

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

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

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

63 *Proof.* [[DF: will continue tomorrow, please don't touch]] Let X'' be any random variable whose
64 support is of size smaller or equal to m , we find a random variable X' with $\text{support}(X') \subseteq$
65 $\text{support}(X)$ and $d_K(X, X') = d_K(X, X'')$. Let $\{x_1, \dots, x_n\} = \text{support}(X)$, and let $x_0 =$
66 $-\infty, x_{n+1} = \infty$. Consider the random variable X' whose probability mass function is $f_{X'}(x_i) =$
67 $P(x_{i-1} < X'' < x_i)$ for $i = 1, \dots, n-1$, $f_{X'}(x_n) = P(x_n - 1 < X'' < x_{n+1})$ and $F_{X'}(x) = 0$
68 if $x \notin \text{support}(X)$. First see that for every $x_i < x < x_{i+1}$ for $i = 0, \dots, n-1$ we have
69 $|F_X(x) - F_{X'}(x)| \leq |F_X(x_i) - F_{X'}(x_i)|$

70 By definition $d_K(X, X') = d_K(X, X'')$. Since $\text{support}(X') \subseteq \text{support}(X)$ and $|\text{support}(X')| <$
71 $|\text{support}(X'')| \leq m$, we get that $d_K(X, X'') \geq \varepsilon$. Thus, ε is a global minimum and any
72 $X' \in \text{argmin}\{d_K(X, X') : \text{support}(X') \subseteq \text{support}(X), |\text{support}(X')| \leq m\}$ is an m -optimal
73 approximation of X . \square

74 Next, note that every random variable X'' with support of size at most m that is contained in
75 $\text{support}(X)$ can be described by first setting the (at most m) elements of the support of X'' ; then for
76 every such option, determine X'' by setting probability values for the elements in the chosen support
77 of X' , and setting 0 for rest of the elements.

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

83 3.1 Step 1: Finding an X' in \mathbb{X}_S that minimizes $d_K(X, X')$

84 We first fix a set $S \subseteq \text{support}(X)$ of size at most m , and among all the random variables in
85 \mathbb{X}_S find one with a minimal distance from X . Denote the elements of S in increasing order by
86 $S = \{x_1 < \dots < x_m\}$ and let $x_0 = -\infty$, and $x_{m+1} = \infty$. For every $1 < i \leq m$ let \hat{x}_i be the
87 maximal element of $\text{support}(X)$ that is smaller than x_i .

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

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

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

91 Note that $x_i = -\infty$ for $i = 0$ and $x_i = \infty$ for $i = m + 1$. Also note that $P(x_i < X < x_{i+1}) =$
92 $F_X(\hat{x}_{i+1}) - F_X(x_i)$, a fact that we will use throughout this section.

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

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

97 The intuition behind choosing these specific weights and $\varepsilon(X, S)$ being a lower bound is as follows.
98 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
99 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
100 have to take into additional account is increased by $F_X(\hat{x}_{i+1}) - F_X(x_i) = P(x_i < X < x_{i+1})$.

101 Formally we have the following.

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

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

106 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
107 $d_K(X, X') \geq |F_X(\hat{x}_{i+1})| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

108 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
109 $d_K(X, X') \geq |1 - F_X(\hat{x}_i)| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

110 Otherwise for every $1 \leq i < m$, we use the fact that $\max\{|a|, |b|\} \geq |a - b|/2$ for every $a, b \in \mathbb{R}$, 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 $d_K(X, X') \geq$
112 $1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_1 < X < x_2)/2 = w(x_i, x_{i+1})$.

113 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
114 follows. \square

115 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
116 m -approximation among the set \mathbb{X}_S . We define X' as follows:

117 **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 $f_{X'}(x) = 0$
118 for $x \notin S$.*

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

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

121 *Proof.* From definition $f_{X'}(x_i) \geq 0$ for every i . To see that $\sum_i f_{X'}(x_i) = 1$, we have $\sum_i f_{X'}(x_i) =$
122 $\sum_i (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = \sum_{x_i \in S} f_X(x_i) + w(x_0, x_1) + \sum_{0 < i < m} 2w(x_i, x_{i+1}) +$
123 $w(x_m, x_{m+1}) = \sum_{x_i \in S} P(X = x_i) + P(x_0 < X < X_1) + \sum_{0 < i < m} P(x_i < X < X_{i+1}) +$
124 $P(x_m < X < X_{m+1}) = 1$ since this sum is the entire cpt of X . \square

125 Note that X' can be constructed in linear time to the size of the cdf of X . Intuitively the setting of
126 X' allows to take an "advantage" of distance from X at the elements of $\text{support}(X')$, to avoid the
127 overall increased distance of X from X' at the elements that are not at $\text{support}(X)$ and in which
128 $f_{X'}$ is set to 0. Formally we have the following.

129 **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 $-w(x_i, x_{i+1}) \leq$
130 $F_X(x) - F_{X'}(x) \leq w(x_i, x_{i+1})$.*

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

132 First see that $F_{X'}(j) = 0$ for every $x_0 < j < x_1$ and therefore $F_X(j) - F_{X'}(j) = F_X(j) - 0 \leq$
133 $F_X(\hat{x}_1) = F_X(\hat{x}_1) - F_X(x_0) = w(x_0, x_1)$. For $j = x_1$ we have $F_X(x_1) - F_{X'}(x_1) = F_X(\hat{x}_1) +$
134 $f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1)) = w(x_0, x_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) +$
135 $f_X(x_1)) = -w(x_1, x_2)$.

136 Next assume that $F_X(\hat{x}_i) - F_{X'}(\hat{x}_i) = w(x_{i-1}, x_i)$. Then $F_X(x_i) - F_{X'}(x_i) = F_X(\hat{x}_i) + f_X(x_i) -$
137 $(w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = w(x_{i-1}, x_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) +$
138 $f_X(x_i)) = -w(x_i, x_{i+1})$.

139 As before we have that for all $x_i < j < x_{i+1}$, we have $F_X(j) - F_{X'}(j) = F_X(j) - F_{X'}(\hat{x}_{i+1}) \leq$
140 $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})$. Then $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1}) = (F_X(x_i) + P(x_i < x < x_{i+1})) -$
141 $F_{X'}(x_i) = -w(x_i, x_{i+1}) + 2w(x_i, x_{i+1}) = w(x_i, x_{i+1})$.

142 Finally for $x_m \leq j \leq x_{m+1}$ we have that $F_{X'}(x_m) = 1$ therefore $F_X(x_m) - F_{X'}(x_m) = (1 -$
143 $P(x_m < X < x_{m+1})) - 1 = P(x_m < X < x_{m+1}) = w(x_m, x_{m+1})$, and for every $x_m < j <$
144 x_{m+1} we have $F_X(j) - F_{X'}(j) < (1 - P(x_m < X < x_{m+1})) - 1 < -P(x_m < X < x_{m+1}) =$
145 $-w(x_m, x_{m+1})$ as required. \square

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

147 **Corrolary 9.** $d_K(X, X') = \varepsilon(X, S)$.

148 3.2 Step 2: Finding an S that minimizes $\varepsilon(X, S)$

149 Chakravarty, Orlin, and Rothblum [2] proposed a polynomial-time method that, given a certain
150 objective functions (additive), finds an optimal consecutive partition. Their method involves the
151 construction of a graph such that the (consecutive) set partitioning problem is reduced to the problem
152 of finding the shortest path in that graph.

153 The KolmogorovApprox algorithm (Algorithm 1) starts by constructing a directed weighted graph
154 G similar to the method of Chakravarty, Orlin, and Rothblum [2]. The nodes V consist of the
155 support of X together with an extra two nodes, $-\infty$ and ∞ for technical reasons, whereas the
156 edges E connect every pair of nodes in one direction (lines 1-2). The weight w of each edge
157 $e = (x, y) \in E$ is determined by one of two cases as in Definition 3. The values taken are
158 non inclusive, since we are interested only in the error value. The source node of the shortest
159 path problem at hand corresponds to the $-\infty$ node added to G in the construction phase, and
160 the target node is the extra node ∞ . The set of all solution paths in G , i.e., those starting at
161 $-\infty$ and ending in ∞ with at most m edges, is called $paths(G, -\infty, \infty)$. The goal is to find the
162 path l in $paths(G, -\infty, \infty)$ with the lightest bottleneck (line 3). This can be achieved by using
163 the *Bellman – Ford* algorithm with two tweaks. The first is to iterate the graph G in order to
164 find only paths with length of at most m edges. The second is to find the lightest bottleneck as
165 opposed to the traditional objective of finding the shortest path. This is performed by modifying the
166 manner of “relaxation” to $bottleneck(x) = \min[\max(bottleneck(v), w(e))]$, done also in [9, 15].
167 Consequently, we find the lightest maximal edge in a path of length $\leq m$, which represents the
168 minimal error, $\varepsilon(X, S)$, defined in Definition 4 where the nodes in path l represent the elements in set
169 S . The approximated random variable X' is then derived from the resulting path l (lines 4-5). Every
170 node $x \in l$ represent a value in the new calculated random variable X' , we than iterate the path l to
171 find the probability of the event $f_{X'}(x)$ 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  $(x_0, \dots, x_{m+1}) = l \in \text{argmin}_{l \in paths(G, -\infty, \infty), |l| \leq m} \max\{w(e) : e \in l\}$ 
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  $X'$ 

```

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

173 *Proof.* If we consider the vertexes $S = l \setminus \{-\infty, \infty\}$ for a path $l \in \text{paths}(G, -\infty, \infty)$ we have
 174 that $\max\{w(e) : e \in l\} = \varepsilon(X, S)$. Therefore, line 3 of the algorithm essentially computes a set
 175 $S \in \arg\min_{S \subseteq \text{support}(X), |S| \leq m} \varepsilon(X, S)$. By Corollary 9, the variable X' constructed in lines 4 and
 176 5 satisfies $d_K(X, X') = \varepsilon(X, S)$ and by the minimality of S and by Proposition 5, it is an optimal
 177 approximation. \square

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

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

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

186 Since G is DAG (directed acyclic graph) finding a shortest path takes $O(E + V)$. We only need
 187 to find paths of length $\leq m$, which takes $O(m(E + V))$. Deriving the new random variable X'
 188 from the computed path l takes $O(m)$. For every node x_i in l (at most m nodes), use the already
 189 calculated weights to find the probability mass function $f_{X'}(x_i)$. To conclude, the worst case run-time
 190 complexity is $O(n^2 + mn^2 + m) = O(mn^2)$ and memory complexity is $O(E + V) = O(n^2)$. \square

191 4 A case study and experimental results

192 The case study examined in our experiments is the problem of task trees with deadlines [4, 3].
 193 Hierarchical planning is a well-established field in AI [5, 6, 7], and is still relevant nowadays [1, 16].
 194 A hierarchical plan is a method for representing problems of automated planning in which the
 195 dependency among tasks can be given in the form of networks, here we focus on hierarchical plans
 196 represented by task trees. The leaves in a task tree are *primitive* actions (or tasks), and the internal
 197 nodes are either *sequence* or *parallel* actions. The plans we deal with are of stochastic nature, and
 198 the task duration is described as probability distributions in the leaf nodes. We assume that the
 199 distributions are independent but *not* necessarily identically distributed and that the random variables
 200 are discrete and have a finite support.

201 A sequence node denotes a series of tasks that should be performed consecutively, whereas a parallel
 202 node denotes a set of tasks that begin at the same time. A *valid* plan is one that is fulfilled before some
 203 given *deadline*, i.e., its *makespan* is less than or equal to the deadline. The objective in this context is
 204 to compute the probability that a given plan is valid, or more formally computing $P(X < T)$, where
 205 X is a random variable representing the makespan of the plan and T is the deadline. The problem of
 206 finding the probability that a task tree satisfies a deadline is known to be NP-hard. In fact, even the
 207 problem of summing a set of random variables is NP-hard [11]. This is an example of an explicitly
 208 given random variable that we need to estimate deadline meeting probabilities for.

209 The first experiment we focus on is the problem of task trees with deadlines, and consider three
 210 types of task trees. The first type includes logistic problems of transporting packages by trucks and
 211 airplanes (from IPC2 <http://ipc.icaps-conference.org/>). Hierarchical plans of those logistic problems
 212 were generated by the JSHOP2 planner [12], one parallel node with all descendant task nodes being
 213 in sequence. The second type consists of task trees used as execution plans for the ROBIL team
 214 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 [3] and the Trim [4] 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 $s = 10^6$.

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	0.0009
	4	0.008	0.024	0.04	0.008	0.0016
	10	0	0.028	0.06	0.0117	0.001

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

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 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 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 from a uniform distribution and then normalizing these probabilities so that they sum to one.

Figure 1 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

247 methods. However, errors produced by the KolmogorovApprox are significantly smaller, a half of
 248 the error produced by OptTrim and Trim.

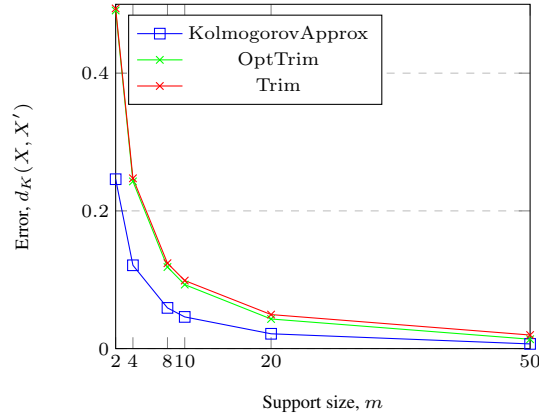


Figure 1: Error comparison between KolmogorovApprox, OptTrim, and Trim, on randomly generated random variables as function of m .

249 We also examined how our algorithm compares to linear programming as described and discussed, for
 250 example, in [13]. We ran an experiment to compare the run-time between the KolmogorovApprox
 251 algorithm with the run-time of a state-of-art implementation of linear programming. We used the
 252 “Minimize” function of Wolfram Mathematica and fed it with the equations $\min_{\alpha \in \mathbb{R}^n} \|x - \alpha\|_\infty$
 253 subject to $\|\alpha\|_0 \leq m$ and $\|\alpha\|_1 = 1$. The run-time comparison results were clear and persuasive,
 254 for a random variable with support size $n = 10$ and $m = 5$, the LP algorithm run-time was 850
 255 seconds, where the KolmogorovApprox algorithm run-time was less than a tenth of a second. For
 256 $n = 100$ and $m = 5$, the KolmogorovApprox algorithm run-time was 0.14 seconds and the LP
 257 algorithm took more than a day. Due to these timing results of the LP algorithm we did not proceed
 258 to examine it any further. Since it is not trivial to formally analyze the run-time of the LP algorithm,
 259 we conclude by the reported experiment that in this case the LP algorithm might not be as efficient as
 260 KolmogorovApprox algorithm whose complexity is proven to be polynomial in Theorem 11.

261 5 Discussion

262 References

- 263 [1] R. Alford, V. Shivashankar, M. Roberts, J. Frank, and D. W. Aha. Hierarchical planning:
 264 Relating task and goal decomposition with task sharing. In *IJCAI*, pages 3022–3029, 2016.
- 265 [2] A. Chakravarty, J. Orlin, and U. Rothblum. A partitioning problem with additive objective with
 266 an application to optimal inventory groupings for joint replenishment. *Operations Research*,
 267 30(5):1018–1022, 1982.
- 268 [3] L. Cohen, T. Grinshpoun, and G. Weiss. Optimal approximation of random variables for
 269 estimating the probability of meeting a plan deadline. In *Proceedings of the Thirty-Second AAAI*
 270 *Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.
- 271 [4] L. Cohen, S. E. Shimony, and G. Weiss. Estimating the probability of meeting a deadline in
 272 hierarchical plans. In *IJCAI*, pages 1551–1557, 2015.
- 273 [5] T. Dean, R. J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and
 274 resources. *Computational Intelligence*, 4(3):381–398, 1988.

- 275 [6] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*,
276 volume 94, pages 1123–1128, 1994.
- 277 [7] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. *Annals of Mathematics*
278 *and Artificial Intelligence*, 18(1):69–93, 1996.
- 279 [8] J. D. Gibbons and S. Chakraborti. Nonparametric statistical inference. In *International*
280 *encyclopedia of statistical science*, pages 977–979. Springer, 2011.
- 281 [9] R. Guérin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM*
282 *Transactions on Networking (TON)*, 10(5):613–620, 2002.
- 283 [10] A. C. Miller and T. R. Rice. Discrete approximations of probability distributions. *Management*
284 *Science*, 29(3):352–362, 1983.
- 285 [11] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. *Computational*
286 *Discrete Mathematics*, pages 79–97, 2001.
- 287 [12] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An
288 HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.
- 289 [13] K. Pavlikov and S. Uryasev. CVaR distance between univariate probability distributions and
290 approximation problems. Technical Report 2015-6, University of Florida, 2016.
- 291 [14] A. N. Pettitt and M. A. Stephens. The kolmogorov-smirnov goodness-of-fit statistic with
292 discrete and grouped data. *Technometrics*, 19(2):205–210, 1977.
- 293 [15] E. Shufan, H. Ilani, and T. Grinshpoun. A two-campus transport problem. In *MISTA*, pages
294 173–184, 2011.
- 295 [16] Z. Xiao, A. Herzig, L. Perrussel, H. Wan, and X. Su. Hierarchical task network planning with
296 task insertion and state constraints. In *IJCAI*, pages 4463–4469, 2017.