
An algorithm for Kolmogorov distance based approximation of discrete random variables

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 whose support (set of possible outcomes) is of
3 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 considered, 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 commonly used for com-
18 paring random variables in statistical practice and literature. Given two random variables X and
19 X' whose cumulative distribution functions (cdfs) are F_X and $F_{X'}$, respectively, the Kolmogorov
20 distance between X and X' is $d_K(X, X') = \sup_t |F_X(t) - F_{X'}(t)|$ (see, e.g., [9]). We say that X'
21 is a good approximation of X 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 in plans and
 37 compare it to alternative 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 type of approximation is for cases where overestimation of the probability of missing
 46 a deadline is acceptable but underestimation is not. In Section 4, we consider the same examples
 47 examined by Cohen et. al. and show how the algorithm proposed in this paper performs relative to
 48 the algorithms proposed there when both over- and under- estimations are allowed. As expected, the
 49 Kolmogorov distance between the approximation and the original random variable is smaller by a
 50 factor of one half, on average, when using the algorithm proposed here.

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

72 The present study is also related to the work of Pavlikov and Uryasev [15], where a procedure for
 73 producing a random variable X' that optimally approximates a random variable X is presented.
 74 Their approximation scheme, achieved using linear programming, is designed for a different notion
 75 of distance (called CVaR). The new contribution of the present work in this context is that our
 76 method is direct, not using linear programming, thus allowing tighter analysis of time and memory
 77 complexity. Also, our method is designed for optimizing the Kolmogorov distance that is more
 78 prevalent in applications. For comparison, in Section 4 we briefly discuss the performance of linear
 79 programming approach similar to the one proposed in [15] for the Kolmogorov distance and compare
 80 it to the algorithm proposed in this paper.

81 3 An Algorithm for Optimal Approximation

82 In the scope of this section, let X be a given random variable with a finite support of size n , and
 83 let $0 < m \leq n$ be a given complexity bound. The section evolves by developing notations and by
 84 collecting facts towards an algorithm for finding an optimal m -approximation of X .

85 The first useful fact is that it is enough to limit our search to approximations X' 's such that
 86 $\text{support}(X') \subseteq \text{support}(X)$:

87 **Lemma 2.** *For every random variable X'' there is a random variable X' such that $\text{support}(X') \subseteq$
 88 $\text{support}(X)$ and $d_K(X, X') \leq d_K(X, X'')$.*

89 *Proof.* Let $\{x_1, \dots, x_n\} = \text{support}(X)$, and let $x_0 = -\infty, x_{n+1} = \infty$. Consider the random
 90 variable X' whose probability mass function is $f_{X'}(x_i) = P(x_{i-1} < X'' \leq x_i)$ for $i = 1, \dots, n -$
 91 1 , $f_{X'}(x_n) = P(x_n - 1 < X'' < x_{n+1})$, and $F_{X'}(x) = 0$ if $x \notin \text{support}(X)$. Since X'
 92 "accumulates" the non-zero probabilities of X'' to the support of X , we have that $f_{X'}$ is a probability
 93 mass function and therefore X' is well defined.

94 First see by construction that $|F_X(x_i) - F_{X'}(x_i)| = |F_X(x_i) - F_{X''}(x_i)|$ for every $1 \leq i \leq n - 1$.
 95 For $i = n$ we have $|F_X(x_n) - F_{X'}(x_n)| = |1 - 1| = 0$. Finally see that $|F_X(x) - F_{X'}(x)| =$
 96 $|F_X(x_i) - F_{X'}(x_i)|$ for every $0 \leq i < n + 1$ and $x_i < x < x_{i+1}$. Therefore we have that
 97 $d_K(X, X') = \max_i |F_X(x_i) - F_{X'}(x_i)| \leq \max_i |F_X(x_i) - F_{X''}(x_i)| \leq d_K(X, X'')$. \square

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

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

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

108 We first fix a set $S \subseteq \text{support}(X)$ of size at most m , and among all the random variables in
 109 \mathbb{X}_S find one with a minimal distance from X . Denote the elements of S in increasing order by
 110 $S = \{x_1 < \dots < x_m\}$ and let $x_0 = -\infty$, and $x_{m+1} = \infty$. For every $1 \leq i \leq m$ let \hat{x}_i be the
 111 maximal element of $\text{support}(X)$ that is smaller than x_i .

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

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

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

115 Note that $x_i = -\infty$ for $i = 0$ and $x_i = \infty$ for $i = m + 1$. Also note that $P(x_i < X < x_{i+1}) =$
 116 $F_X(\hat{x}_{i+1}) - F_X(x_i)$, a fact that we will use throughout this section.

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

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

121 The intuition behind choosing these specific weights and $\varepsilon(X, S)$ being a lower bound is as follows.
 122 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
 123 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
 124 have to take into additional account is increased by $F_X(\hat{x}_{i+1}) - F_X(x_i) = P(x_i < X < x_{i+1})$.

125 Formally we have the following.

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

127 *Proof.* By definition, for every $0 \leq i \leq m$, $d_K(X, X') \geq \max\{|F_X(\hat{x}_{i+1}) -$
 128 $F_{X'}(\hat{x}_{i+1})|, |F_X(x_i) - F_{X'}(x_i)|\}$. Note that $F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i)$ since the probability values
 129 for all the elements not in S are set to 0.

130 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
 131 $d_K(X, X') \geq |F_X(\hat{x}_{i+1})| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

132 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
 133 $d_K(X, X') \geq |1 - F_X(\hat{x}_i)| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

134 Otherwise for every $1 \leq i < m$, we use the fact that $\max\{|a|, |b|\} \geq |a - b|/2$ for every $a, b \in$
 135 \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 $d_K(X, X') \geq$
 136 $1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_1 < X < x_2)/2 = w(x_i, x_{i+1})$.

137 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
 138 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. $d_K(X, X') = \varepsilon(X, S)$.

Step 2: Finding a set S that minimizes $\varepsilon(X, S)$

Finding the m -optimal random variable with a distance $\varepsilon(X, S)$ among the set \mathbb{X}_S , we next proceed to find an S that minimizes $\varepsilon(X, S)$. To obtain that we use a dynamic programming approach, motivated by a method described in [3]. Intuitively we construct a directed graph with a source and a target, in which every (acyclic) source-target path of size $\leq m$ corresponds to a possible support

178 set of the same size, and the weights along that path correspond to the weight as defined in Definition
 179 3. Thus the problem of finding the S that minimizes $\varepsilon(X, S)$ is reduced to the problem of finding
 180 the source-target path \vec{p} of size $\leq m$ in that graph such that the maximal weight of an edge in \vec{p} is
 181 minimal among all other such maximal edges in all other paths of size $\leq m$.

182 In details, we initially constructs a directed weighted graph $G = (V, E, s, t, w)$, where $V =$
 183 v_0, \dots, v_{n+1} correspond to the support of X with the additional $v_0 = s$ as the source vertex and
 184 $v_{n+1} = t$ as the target vertex. We set an edge (v_i, v_j) for every $i < j$ and a weight that is defined
 185 as follows. If $1 < i < j < n$ then $w(v_i, v_j) = P(x_i < X < x_{i+1})/2$. For $i = 0$ or $j = n + 1$ we
 186 set $w(v_i, v_j) = P(x_i < X < x_{i+1})$. Thus the weight of every edge (i, j) corresponds to Definition
 187 3 for a case in which x_i, x_j where consecutive elements in an optional set S of size $\leq m$. Note that
 188 there is a 1-1 correspondence between a subset S of $\text{support}(X)$ of size m , and an $s-t$ path \vec{p}_S in G
 189 where every $x_i \in S$ corresponds to the vertex $v_i \in \vec{p}_S$. On that sense note that the maximal weight
 190 of an edge on \vec{p}_S is identical to $\varepsilon(X, S)$. We denote this maximal weight of an edge by $w(\vec{p}_S)$, and
 191 denote the set of all acyclic s -to- t paths in G with at most m edges is called $\text{paths}_m(G, s, t)$. Thus,
 192 the problem of finding the set S with the minimal $\varepsilon(X, S)$ is trivially reduced to the problem of find-
 193 ing a path $\vec{p} \in \text{paths}_m(G, s, t)$ such that $w(\vec{p})$ is minimal among all $\{w(\vec{p}') | \vec{p}' \in \text{paths}_m(G, s, t)\}$.
 194 This problem can be solved by the standard Bellman-Ford algorithm in m iterations (e.g. [10, 18])
 195 denoted by $\text{BellmanFord}(G, m)$. We thus have the following from the standard Bellman-Ford
 196 technique.

197 **Corollary 10.** *$\text{BellmanFord}(G, m)$ returns a path $\vec{p} \in \text{paths}_m(G, s, t)$ such that $w(\vec{p})$ is minimal*
 198 *among all $\{w(\vec{p}') | \vec{p}' \in \text{paths}_m(G, s, t)\}$.*

199 Step 3: Constructing the overall algorithm

200 We combine Step 1 and Step 2 in the following algorithm called KolmogorovApprox (Algorithm 1)
 201 that follows naturally from the two steps. Given X and $\text{support}(X)$ we add x_0, x_{n+1} and construct
 202 the graph (line 2) as in Step 2. Then we execute the Bellman-Ford algorithm on G for m iterations to
 203 obtained a path $\vec{p} = (v_0, \dots, v_{m+1})$ (line 2) as described in Corollary 10. Finally we use Definition
 204 6 to construct X' from the weights of \vec{p} (lines 4-5).

Algorithm 1: KolmogorovApprox(X, m)

```

1 support( $X$ )  $\cup \{x_0, x_{n+1}\}$ 
2 Construct  $G = (V, E, s, t, w)$  according to Step 2
3  $\vec{p} = (v_0, \dots, v_{m+1}) = \text{Bellman} - \text{Ford}(G, m)$ 
4 for  $0 < i < m + 1$  do
5    $f_{X'}(x_i) = w(v_{i-1}, v_i) + w(v_i, v_{i+1}) + f_X(x_i)$ 
6 return  $X'$ 

```

205 **Theorem 11.** *KolmogorovApprox returns an m -optimal-approximation of X .*

206 *Proof.* From Corollary 10 and the construction of G we get that the path \vec{p} obtained in line 4 of
 207 KolmogorovApprox describes a set S of support os size at most m for which ε_S, X is minimal.
 208 Then from Definition 6 and Corollary 9 we construct X' in lines 4-5 of KolmogorovApprox such
 209 that $d_K(X, X') = \varepsilon(X, S)$. Therefore X' is an m -approximation among all random variables with
 210 support contained in $\text{support}(X)$. Finally from Lemma 2 we have that X' is m -approximation
 211 among all random variables os support of size at most m , thus X' is an m -optimal-approximation
 212 of X . \square

213 Finally we analyze the complexity of KolmogorovApprox as follows.

214 **Theorem 12.** *The KolmogorovApprox(X, m) algorithm runs in time $O(mn^2)$, using $O(n^2)$ mem-*
 215 *ory where $n = |\text{support}(X)|$.*

216 *Proof.* Constructing the graph G as described in Step 2 takes $O(n^2)$. The number of edges in G
 217 is $O(|E|) = O(n^2)$, for every edge the weight is at most the sum of all probabilities between the
 218 source node s and the target node t , which can be calculated efficiently by aggregating the weights
 219 of already calculated edges. The construction is also the only stage that requires memory allocation,
 220 specifically $O(|E| + |V|) = O(n^2)$. Next using the Bellman-Ford algorithm on G for m iterations
 221 takes $O(m(|E| + |V|)) \approx O(mn^2)$. [[DF: cite Corman or some algorithms book]]. Finally deriving
 222 the new random variable X' from the computed path \vec{p} takes $O(m)$ time: For every node x_i in \vec{p}
 223 (at most m nodes), use the already calculated weights to find the probability mass function $f_{X'}(x_i)$.
 224 To conclude, the time complexity of KolmogorovApprox(X, m) is $O(n^2 + mn^2 + m) = O(mn^2)$
 225 and memory complexity is $O(E + V) = O(n^2)$.

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

227

□

228 4 A case study and experimental results

229 The case study examined in our experiments is the problem of task trees with deadlines [5, 4].
 230 Hierarchical planning is a well-established field in AI [6, 7, 8], and is still relevant nowadays [1,
 231 20]. A hierarchical plan is a method for representing problems of automated planning in which
 232 the dependency among tasks can be given in the form of networks, here we focus on hierarchical
 233 plans represented by task trees. The leaves in a task tree are *primitive* actions (or tasks), and the
 234 internal nodes are either *sequence* or *parallel* actions. The plans we deal with are of stochastic
 235 nature, and the task duration is described as probability distributions in the leaf nodes. We assume
 236 that the distributions are independent but *not* necessarily identically distributed and that the random
 237 variables are discrete and have a finite support.

238 A sequence node denotes a series of tasks that should be performed consecutively, whereas a parallel
 239 node denotes a set of tasks that begin at the same time. A *valid* plan is one that is fulfilled before
 240 some given *deadline*, i.e., its *makespan* is less than or equal to the deadline. The objective in this
 241 context is to compute the probability that a given plan is valid, or more formally computing $P(X <$
 242 $T)$, where X is a random variable representing the makespan of the plan and T is the deadline. The
 243 problem of finding the probability that a task tree satisfies a deadline is known to be NP-hard. In
 244 fact, even the problem of summing a set of random variables is NP-hard [13]. This is an example of
 245 an explicitly given random variable that we need to estimate deadline meeting probabilities for.

246 The first experiment we focus on is the problem of task trees with deadlines, and consider three
 247 types of task trees. The first type includes logistic problems of transporting packages by trucks and
 248 airplanes (from IPC2 <http://ipc.icaps-conference.org/>). Hierarchical plans of those logistic problems
 249 were generated by the JSHOP2 planner [14], one parallel node with all descendant task nodes being
 250 in sequence. The second type consists of task trees used as execution plans for the ROBIL team
 251 entry in the DARPA robotics challenge (DRC simulation phase), and the third type is of linear plans
 252 (sequential task trees). The primitive tasks in all the trees are modeled as discrete random variables
 253 with support of size M obtained by discretization of uniform distributions over various intervals.
 254 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 $s = 10^6$.

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.

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 uniformly and then normalizing these probabilities so that they sum to 1.

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 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 programming 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 programming. We used the

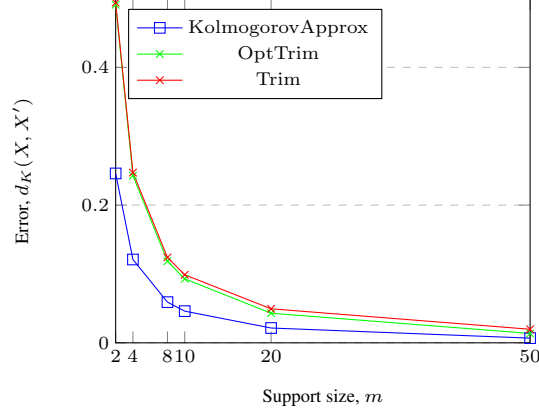


Figure 1: Error comparison between KolmogorovApprox, OptTrim, and Trim, on randomly generated random variables as function of m .

“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 12.

5 Discussion

Compact representations of distributions is mentioned in the literature in various contexts for various applications. In this paper, we are interested in finding optimal approximation of a random variables under the Kolmogorov metric which we define as optimal m -approximation. In order to achieve this optimal approximation two steps were taken, find the support of the optimal random variable and then calculate the pmf of each and every value in that support to minimize the error. Proofs of existences, optimality and run-time were detailed in Section 3 and the main algorithm was presented, the KolmogorovApprox algorithm. Establishing the main contribution of this paper which is to present an optimal approximation scheme and to show it can be achieved in polynomial run-time. Furthermore, empirical evaluation was conducted on different domains and application to examine the algorithm performance in practice. We were interested in two aspects of performance - accuracy and run-time. Regarding to accuracy, as expected, the suggested KolmogorovApprox algorithm results much smaller error compared to the other methods, sometimes, in more then factor of 2. Regarding to run-time, KolmogorovApprox algorithm run-time is significantly much faster then LP approach. However, compared to other approximation methods accuracy vs. run-time is a trade off yet to be examined. Another interesting experiment that can be conducted in future work is to add the presented approach as one of the methods examined in [21] and compare it to the binning approaches.

As elaborated in the paper, our algorithm improves on the approach of Cohen, Shimony and Weiss [5] and [4] in that it finds an optimal two sided Kolmogorov approximation, and not just one sided. We consider this paper as a step in the examination of algorithms for optimal approxi-

319 mations of random variables. Beyond the Kolmogorov measure studied here we believe that similar
 320 approaches may apply also to total variation, the Wasserstein distance, and to other measures of
 321 approximations for other purposes.

322 References

- 323 [1] R. Alford, V. Shivashankar, M. Roberts, J. Frank, and D. W. Aha. Hierarchical planning:
 324 Relating task and goal decomposition with task sharing. In *IJCAI*, pages 3022–3029, 2016.
- 325 [2] C. Bolton et al. *Logistic regression and its application in credit scoring*. PhD thesis, Citeseer,
 326 2010.
- 327 [3] A. Chakravarty, J. Orlin, and U. Rothblum. A partitioning problem with additive objective with
 328 an application to optimal inventory groupings for joint replenishment. *Operations Research*,
 329 30(5):1018–1022, 1982.
- 330 [4] L. Cohen, T. Grinshpoun, and G. Weiss. Optimal approximation of random variables for
 331 estimating the probability of meeting a plan deadline. In *Proceedings of the Thirty-Second*
 332 *AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*,
 333 2018.
- 334 [5] L. Cohen, S. E. Shimony, and G. Weiss. Estimating the probability of meeting a deadline in
 335 hierarchical plans. In *IJCAI*, pages 1551–1557, 2015.
- 336 [6] T. Dean, R. J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and
 337 resources. *Computational Intelligence*, 4(3):381–398, 1988.
- 338 [7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*,
 339 volume 94, pages 1123–1128, 1994.
- 340 [8] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. *Annals of Mathe-*
 341 *matics and Artificial Intelligence*, 18(1):69–93, 1996.
- 342 [9] J. D. Gibbons and S. Chakraborti. Nonparametric statistical inference. In *International ency-*
 343 *clopedia of statistical science*, pages 977–979. Springer, 2011.
- 344 [10] R. Guérin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM Trans-*
 345 *actions on Networking (TON)*, 10(5):613–620, 2002.
- 346 [11] E. Mays. *Handbook of credit scoring*. Global Professional Publishi, 2001.
- 347 [12] A. C. Miller and T. R. Rice. Discrete approximations of probability distributions. *Management*
 348 *Science*, 29(3):352–362, 1983.
- 349 [13] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. *Computa-*
 350 *tional Discrete Mathematics*, pages 79–97, 2001.
- 351 [14] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An
 352 HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.
- 353 [15] K. Pavlikov and S. Uryasev. CVaR distance between univariate probability distributions and
 354 approximation problems. Technical Report 2015-6, University of Florida, 2016.
- 355 [16] A. N. Pettitt and M. A. Stephens. The kolmogorov-smirnov goodness-of-fit statistic with dis-
 356 crete and grouped data. *Technometrics*, 19(2):205–210, 1977.

- 357 [17] M. Refaat. *Credit Risk Scorecard: Development and Implementation Using SAS*. Lulu. com,
358 2011.
- 359 [18] E. Shufan, H. Ilani, and T. Grinshpoun. A two-campus transport problem. In *MISTA*, pages
360 173–184, 2011.
- 361 [19] N. Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*,
362 volume 3. John Wiley & Sons, 2012.
- 363 [20] Z. Xiao, A. Herzig, L. Perrussel, H. Wan, and X. Su. Hierarchical task network planning with
364 task insertion and state constraints. In *IJCAI*, pages 4463–4469, 2017.
- 365 [21] G. Zeng. A comparison study of computational methods of kolmogorov–smirnov statistic in
366 credit scoring. *Communications in Statistics-Simulation and Computation*, 46(10):7744–7760,
367 2017.