

---

# Kolmogorov Approximation

---

Anonymous Author(s)

Affiliation

Address

email

## 1 Introduction

Many different approaches to approximation of probability distributions are studied in the literature [9, 12, 13]. 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,

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

## 36 2 Related Work

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

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

39 where  $\|\alpha\|_0 = |\{i : \alpha_i \neq 0, i = 1, \dots, p\}|$  is the  $\ell_0$  pseudo-norm, counting the number of non-zero  
 40 coordinates of  $\alpha$ . This problem is known to be NP-Hard with a reduction to NP-complete subset  
 41 selection problems.

In these terms, using also the  $\ell_\infty$  norm that represents the maximal coordinate and the  $\ell_1$  norm that represents the sum of the coordinates, our problem can be phrased as:

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

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

## 50 3 An Algorithm for Optimal Approximation

51 Let, in the scope of this section,  $X$  be a given random variable with a finite support of size  $n$ , and let  
 52  $0 < m \leq n$  be a given complexity bound. We describe an algorithm for finding find an  $m$ -optimal  
 53 approximation of  $X$  in steps.

54 The first step is to show that it is enough to limit our search to approximations  $X'$ 's such that  
 55  $\text{support}(X') \subseteq \text{support}(X)$ .

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

57 *Proof.* Let  $X''$  be any  $m$ -optimal approximation of  $X$ . Let  $\{x_1, \dots, x_n\} = \text{support}(X)$ . The  
 58 random variable  $X'$  whose probability mass function is  $f_{X'}(x_i) =$

59

□

60 Next, note that every random variable  $X''$  with support of size at most  $m$  that is contained in  
 61  $\text{support}(X)$  be described by first setting the (at most  $m$ ) elements of the support of  $X''$ ; then for

every such option, determine  $X''$  by setting probability values for the elements in the chosen support of  $X'$ , and setting 0 for rest of the elements.

Since from Lemma 2 we can assume without loss of generality that if  $X'$  is an  $m$ -optimal approximation variable for  $X$  then  $\text{support}(X') \subseteq \text{support}(X)$ , our search to find such  $X'$  takes two steps. Denote the set of random variables with support  $S$  by  $\mathbb{X}_S$ . In step 1, we find the  $m$ -optimal approximation random variable among all random variables in  $\mathbb{X}_S$ , and denote the  $m$ -optimal distance for  $\mathbb{X}_S$  by  $\varepsilon(X, S)$ . Next, in Step 2, among all the possible supports we find the support setting  $S$  of size  $\leq m$  for which  $\varepsilon(X, S)$  is minimal: We describe an efficient way to do so.

### 3.1 Step 1

We first fix a set  $S \subseteq \text{support}(X)$  of size at most  $m$ , and among all the random variables in  $\mathbb{X}_S$  find one with a minimal distance from  $X$ . To that, set  $S = \{x_1 < \dots < x_m\} \subseteq \text{support}(X)$ . To simplify the proofs set  $x_0 = -\infty$ , and  $x_{m+1} = \infty$ . Then  $x_0 < x_1$  and  $x_m < x_{m+1}$ . In addition recall that for every random variable  $X''$   $F_{X''}(-\infty) = 0$  and  $F_{X''}(\infty) = 1$ . Finally, for every  $1 \leq i \leq m$  let  $\hat{x}_i$  be the maximal element of  $\text{support}(X)$  that is smaller than  $x_i$ . For the rest of this section we assume  $S$  is fixed and therefore is not necessarily included in the notation.

Next, as the elements of  $S$  are also elements of  $\text{support}(X)$ , we can define the following weight function that we use to find the  $m$ -optimal distance  $\varepsilon(X, S)$ .

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

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

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

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

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

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

Formally we have the following.

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

*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.

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

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

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  $d_k(X, X') \geq 1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_1 < X < x_2)/2 = w(x_i, x_{i+1})$ .

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

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

We first see that  $X'$  is a properly defined random variable. We then discuss the properties of  $X'$ .

**Lemma 7.**  $X'$  is a proper random variable.

*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  $j \in \text{support}(X)$  and let  $0 \leq i \leq m$  such that  $x_i \leq j \leq x_{i+1}$ . Then  $-w(x_i, x_{i+1}) \leq F_X(j) - F_{X'}(j) \leq w(x_i, x_{i+1})$ .

*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.  $\square$

Finally the  $m$ -optimality of  $X'$  w.r.t.  $\mathbb{X}_S$  is straightforward from Lemma 8 and the definition of  $\varepsilon(X, S)$ . Therefore we have.

[[DF: change to corollary]]

138 **Proposition 9.**  $d_k(X, X') = \varepsilon(X, S)$ .

### 139 3.2 Step 2

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

144 The KolmogorovApprox algorithm (Algorithm 1) starts by constructing a directed weighted graph  
 145  $G$  similar to the method of Chakravarty, Orlin, and Rothblum [2]. The nodes  $V$  consist of the  
 146 support of  $X$  together with an extra two nodes,  $-\infty$  and  $\infty$  for technical reasons, whereas the  
 147 edges  $E$  connect every pair of nodes in one direction (lines 1-2). The weight  $w$  of each edge  
 148  $e = (x, y) \in E$  is determined by one of two cases as in Definition 3. The first is where nodes  
 149  $x$  or  $y$  are the source or target nodes respectively. In this case, the weight is the probability of  $X$   
 150 to get a value between  $x$  and  $y$ , non inclusive, i.e.,  $w(e) = \Pr(x < X < y)$ . The second case  
 151 is where  $x$  and  $y$  are not a source or target nodes, here the weight is the probability of  $X$  to get a  
 152 value between  $x$  and  $y$ , non inclusive, divided by two i.e.,  $w(e) = \Pr(x < X < y)/2$ . The values  
 153 taken are non inclusive, since we are interested only in the error value. The source node of the  
 154 shortest path problem at hand corresponds to the  $-\infty$  node added to  $G$  in the construction phase,  
 155 and the target node is the extra node  $\infty$ . The set of all solution paths in  $G$ , i.e., those starting at  
 156  $-\infty$  and ending in  $\infty$  with at most  $m$  edges, is called  $paths(G, -\infty, \infty)$ . The goal is to find the  
 157 path  $l$  in  $paths(G, -\infty, \infty)$  with the lightest bottleneck (line 3). This can be achieved by using the  
 158 *Bellman – Ford* algorithm with two tweaks. The first is to iterate the graph  $G$  in order to find only  
 159 paths with length of at most  $m$  edges. The second is to find the lightest bottleneck as opposed to  
 160 the traditional objective of finding the shortest path. This is performed by modifying the manner of  
 161 “relaxation” to  $bottleneck(x) = \min[\max(bottleneck(v), w(e))]$ , done also in [14]. Consequently,  
 162 we find the lightest maximal edge in a path of length  $\leq m$ , which represents the minimal error,  
 163  $\varepsilon(X, S)$ , defined in Definition ?? where the nodes in path  $l$  represent the elements in set  $S$ . The  
 164 approximated random variable  $X'$  is then derived from the resulting path  $l$  (lines 4-5). Every node  
 165  $x \in l$  represent a value in the new calculated random variable  $X'$ , we than iterate the path  $l$  to find  
 166 the probability of the event  $f_{X'}(x)$  as described in Definition 6. For every edge  $(x_i, x_j) \in l$  we  
 167 determine: if  $(x_i, x_j)$  is the first edge in the path  $l$  (i.e.  $x_i = -\infty$ ), then node  $x_j$  gets the full weight  
 168  $w(x_i, x_j)$  and it's own weight in  $X$  such that  $f_{X'}(x_j) = f_X(x_j) + w(x_i, x_j)$ . If  $(x_i, x_j)$  in not the  
 169 first nor the last edge in path  $l$  then we divide it's weight between nodes  $x_i$  and  $x_j$  in addition to their  
 170 own original weight in  $X$  and the probability that already accumulated. If  $(x_i, x_j)$  is the last edge  
 171 in the path  $l$  (i.e.  $x_j = \infty$ ) then node  $x_i$  gets the full weight  $w(x_i, x_j)$  in addition to what was already  
 172 accumulated such that  $f_{X'}(x_j) = f_{X'}(x_i) + w(x_i, x_j)$ .

---

**Algorithm 1:** KolmogorovApprox( $X, m$ )

---

```

1  $S = \text{support}(X) \cup \{\infty, -\infty\}$ 
2  $G = (V, E) = (S, \{(x, y) : x < y\})$ 
3  $(x_0, \dots, x_{m+1}) = l = \text{argmin}_{l \in paths(G, -\infty, \infty), |l| \leq m} \max\{w(e) : e \in l\}$ 
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  $X'$ 

```

---

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

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

176 *Proof.* Constructing the graph  $G$  takes  $O(n^2)$ . The number of edges is  $O(E) \approx O(n^2)$  and for every  
 177 edge the weight is at most the sum of all probabilities between the source node  $-\infty$  and the target  
 178 node  $\infty$ , which can be done efficiently by aggregating the weights of already calculated edges. The  
 179 construction is also the only stage that requires memory allocation, specifically  $O(E + V) = O(n^2)$ .  
 180 Finding the shortest path takes  $O(m(E + V)) \approx O(mn^2)$ . Since  $G$  is DAG (directed acyclic graph)  
 181 finding shortest path takes  $O(E + V)$ . We only need to find paths of length  $\leq m$ , which takes  
 182  $O(m(E + V))$ . Deriving the new random variable  $X'$  from the computed path  $l$  takes  $O(mn)$ . For  
 183 every node in  $l$  (at most  $m$  nodes), calculating the probability  $P(s < X < \infty)$  takes at most  $n$ .  
 184 To conclude, the worst case run-time complexity is  $O(n^2 + mn^2 + mn) = O(mn^2)$  and memory  
 185 complexity is  $O(E + V) = O(n^2)$ .  $\square$

## 186 4 A case study and experimental results

187 The case study examined in our experiments is the problem of task trees with deadlines [4, 3].  
 188 Hierarchical planning is a well-established field in AI [5, 6, 7], and is still relevant nowadays [1, 15].  
 189 A hierarchical plan is a method for representing problems of automated planning in which the  
 190 dependency among tasks can be given in the form of networks, here we focus on hierarchical plans  
 191 represented by task trees. The leaves in a task tree are *primitive* actions (or tasks), and the internal  
 192 nodes are either *sequence* or *parallel* actions. The plans we deal with are of stochastic nature, where  
 193 the duration of a primitive action is given by a random variable.

194 A sequence node denotes a series of tasks that should be performed consecutively, whereas a parallel  
 195 node denotes a set of tasks that begin at the same time. A *valid* plan is one that is fulfilled before some  
 196 given *deadline*, i.e., its *makespan* is less than or equal to the deadline. The objective in this context  
 197 is to compute the probability that a given plan is valid, or more formally computing  $P(X < T)$ ,  
 198 where  $X$  is a random variable representing the makespan of the plan and  $T$  is the deadline. As said  
 199 above, resource consumption (task duration) is uncertain, and described as probability distributions  
 200 in the leaf nodes. We assume that the distributions are independent but *not* necessarily identically  
 201 distributed and that the random variables are discrete and have a finite support.

202 The problem of finding the probability that a task tree satisfies a deadline is known to be NP-hard. In  
 203 fact, even the problem of summing a set of random variables is NP-hard [10]. This is an example of  
 204 an explicitly given random variable that we need to estimate deadline meeting probabilities for.

205 In the first experiment we focus on is the problem of task trees with deadlines, and consider three  
 206 types of task trees. The first type includes logistic problems of transporting packages by trucks and  
 207 airplanes (from IPC2 <http://ipc.icaps-conference.org/>). Hierarchical plans of those logistic problems  
 208 were generated by the JSHOP2 planner [11] (see example problem, Figure 1, one parallel node with  
 209 all descendant task nodes being in sequence). The second type consists of task trees used as execution  
 210 plans for the ROBIL team entry in the DARPA robotics challenge (DRC simulation phase), and the  
 211 third type is of linear plans (sequential task trees). The primitive tasks in all the trees are modeled as  
 212 discrete random variables with support of size  $M$  obtained by discretization of uniform distributions  
 213 over various intervals. The number of tasks in a tree is denoted by  $N$ .

214 We implemented the approximation algorithm for solving the deadline problem with four different  
 215 methods of approximation. The first two are for achieving a one-sided Kolmogorov approximation –  
 216 the OptTrim [3] and the Trim [4] operators, and the third is a simple sampling scheme. We used those

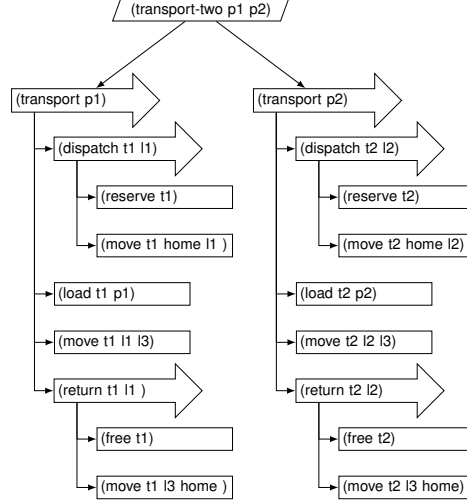


Figure 1: A plan generated by the JSHOP2 algorithm. Arrow shapes represent sequence nodes, parallelograms represent parallel nodes, and rectangles represent primitive nodes.

217 methods as a comparison to the Kolmogorov approximation with the suggested KolmogorovApprox  
 218 algorithm. The parameter  $m$  of OptTrim and KolmogorovApprox corresponds to the inverse of  $\varepsilon$   
 219 given to the Trim operator. Note that in order to obtain some error  $\varepsilon$ , one must take into consideration  
 220 the size of the task tree  $N$ , therefore,  $m/N = 1/(\varepsilon \cdot N)$ . We ran also an exact computation as  
 221 a reference to the approximated one in order to calculate the error. The experiments conducted  
 222 with the following operators and their parameters: KolmogorovApprox operator with  $m = 10 \cdot N$ ,  
 223 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	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
	4	0	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.

224

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

232 In order to better understand the quality gaps in practice between KolmogorovApprox, OptTrim,  
 233 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 2 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, safe to say, a half of the error produced by OptTrim and Trim, it is clear both in the table (the relative error is mostly above 1) and in the graph.

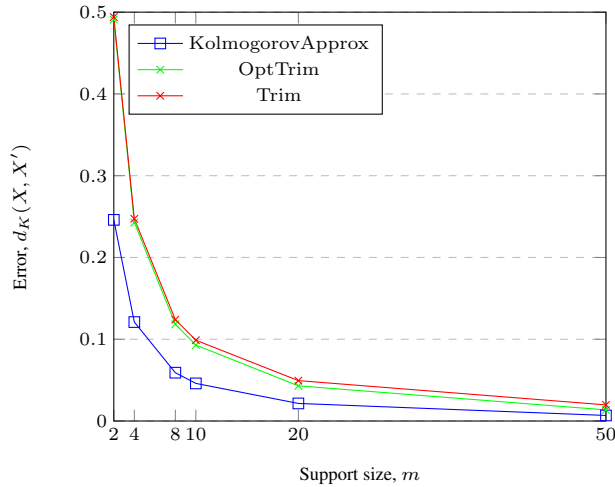


Figure 2: Error comparison between KolmogorovApprox, OptTrim, and Trim, on randomly generated random variables as function of  $m$ .

We also examined how our algorithm compares to linear programming as described and discussed, for example, in [12]. 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 “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] 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.
- [3] 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.
- [4] L. Cohen, S. E. Shimony, and G. Weiss. Estimating the probability of meeting a deadline in hierarchical plans. In *IJCAI*, pages 1551–1557, 2015.
- [5] T. Dean, R. J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4(3):381–398, 1988.
- [6] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.
- [7] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93, 1996.
- [8] J. D. Gibbons and S. Chakraborti. Nonparametric statistical inference. In *International encyclopedia of statistical science*, pages 977–979. Springer, 2011.
- [9] A. C. Miller and T. R. Rice. Discrete approximations of probability distributions. *Management Science*, 29(3):352–362, 1983.
- [10] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. *Computational Discrete Mathematics*, pages 79–97, 2001.
- [11] 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.
- [12] K. Pavlikov and S. Uryasev. CVaR distance between univariate probability distributions and approximation problems. Technical Report 2015-6, University of Florida, 2016.
- [13] 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.
- [14] E. Shufan, H. Ilani, and T. Grinshpoun. A two-campus transport problem. In *MISTA*, pages 173–184, 2011.
- [15] Z. Xiao, A. Herzig, L. Perrussel, H. Wan, and X. Su. Hierarchical task network planning with task insertion and state constraints. In *IJCAI*, pages 4463–4469, 2017.