
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 ℓ_0 pseudo-norm, counting the number of non-zero
 40 coordinates of α . This problem is known to be NP-Hard with a reduction to NP-complete subset
 41 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:

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

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 α is related to X' such that the i th coordinate of α 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 In the scope of this section, let 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 first develop notations and collect facts towards an
 53 algorithm for finding an optimal m -approximation of X .

54 The first useful fact is 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 optimal m -approximation X' of X such that $\text{support}(X') \subseteq \text{support}(X)$.*

57 *Proof.* [[DF: will continue tomorrow, please don't touch]] Let X'' be any random variable whose
 58 support is of size smaller or equal to m , we find a random variable X' with $\text{support}(X') \subseteq$
 59 $\text{support}(X)$ and $d_K(X, X') = d_K(X, X'')$. Let $\{x_1, \dots, x_n\} = \text{support}(X)$, and let $x_0 =$
 60 $-\infty, x_{n+1} = \infty$. Consider the the random variable X' whose probability mass function is $f_{X'}(x_i) =$
 61 $P(x_{i-1} < X'' < x_i)$ for $i = 1, \dots, n-1$, $f_{X'}(x_n) = P(x_n - 1 < X'' < x_{n+1})$ and $F_{X'}(x) = 0$

62 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
63 $|F_X(x) - F_{X'}(x)| \leq |F_X(x_i) - F_{X'}(x_i)|$
64 By definition $d_K(X, X') = d_K(X, X'')$. Since $\text{support}(X') \subseteq \text{support}(X)$ and $|\text{support}(X')| <$
65 $|\text{support}(X'')| \leq m$, we get that $d_K(X, X'') \geq \varepsilon$. Thus, ε is a global minimum and any
66 $X' \in \text{argmin}\{d_K(X, X') : \text{support}(X') \subseteq \text{support}(X), |\text{support}(X')| \leq m\}$ is an m -optimal
67 approximation of X . \square

68 Next, note that every random variable X'' with support of size at most m that is contained in
69 $\text{support}(X)$ can be described by first setting the (at most m) elements of the support of X'' ; then for
70 every such option, determine X'' by setting probability values for the elements in the chosen support
71 of X' , and setting 0 for rest of the elements.

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

77 3.1 Step 1: Finding an X' in \mathbb{X}_S that minimizes $d_K(X, X')$

78 We first fix a set $S \subseteq \text{support}(X)$ of size at most m , and among all the random variables in
79 \mathbb{X}_S find one with a minimal distance from X . Denote the elements of S in increasing order by
80 $S = \{x_1 < \dots < x_m\}$ and let $x_0 = -\infty$, and $x_{m+1} = \infty$. For every $1 \leq i \leq m$ let \hat{x}_i be the
81 maximal element of $\text{support}(X)$ that is smaller than x_i .

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

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

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

85 Note that $x_i = -\infty$ for $i = 0$ and $x_i = \infty$ for $i = m+1$. Also note that $P(x_i < X < x_{i+1}) =$
86 $F_X(\hat{x}_{i+1}) - F_X(x_i)$, a fact that we will use throughout this section.

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

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

91 The intuition behind choosing these specific weights and $\varepsilon(X, S)$ being a lower bound is as follows.
92 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
93 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
94 have to take into additional account is increased by $F_X(\hat{x}_{i+1}) - F_X(x_i) = P(x_i < X < x_{i+1})$.

95 Formally we have the following.

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

97 *Proof.* By definition, for every $0 \leq i \leq m$, $d_k(X, X') \geq \max\{|F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})|, |F_X(x_i) -$
 98 $F_{X'}(x_i)|\}$. Note that $F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i)$ since the probability values for all the elements not in S
 99 are set to 0.

100 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
 101 $d_k(X, X') \geq |F_X(\hat{x}_{i+1})| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

102 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
 103 $d_k(X, X') \geq |1 - F_X(\hat{x}_i)| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

104 Otherwise for every $1 \leq i < m$, we use the fact that $\max\{|a|, |b|\} \geq |a - b|/2$ for every $a, b \in$
 105 \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 $d_k(X, X') \geq$
 106 $1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_1 < X < x_2)/2 = w(x_i, x_{i+1})$.

107 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
 108 follows. \square

109 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
 110 m -approximation among the set \mathbb{X}_S . We define X' as follows:

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

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

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

115 *Proof.* From definition $f_{X'}(x_i) \geq 0$ for every i . To see that $\sum_i f_{X'}(x_i) = 1$, we have $\sum_i f_{X'}(x_i) =$
 116 $\sum_i (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = \sum_{x_i \in S} f_X(x_i) + w(x_0, x_1) + \sum_{0 < i < m} 2w(x_i, x_{i+1}) +$
 117 $w(x_m, x_{m+1}) = \sum_{x_i \in S} P(X = x_i) + P(x_0 < X < x_1) + \sum_{0 < i < m} P(x_i < X < x_{i+1}) +$
 118 $P(x_m < X < x_{m+1}) = 1$ since this sum is the entire cpt of X . \square

119 Note that X' can be constructed in linear time to the size of the cdf of X . Intuitively the setting of
 120 X' allows to take an "advantage" of distance from X at the elements of $\text{support}(X')$, to avoid the
 121 overall increased distance of X from X' at the elements that are not at $\text{support}(X)$ and in which
 122 $f_{X'}$ is set to 0. Formally we have the following.

123 **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 $-w(x_i, x_{i+1}) \leq$
 124 $F_X(x) - F_{X'}(x) \leq w(x_i, x_{i+1})$.

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

126 First see that $F_{X'}(j) = 0$ for every $x_0 < j < x_1$ and therefore $F_X(j) - F_{X'}(j) = F_X(j) - 0 \leq$
 127 $F_X(\hat{x}_1) = F_X(\hat{x}_1) - F_X(x_0) = w(x_0, x_1)$. For $j = x_1$ we have $F_X(x_1) - F_{X'}(x_1) = F_X(\hat{x}_1) +$
 128 $f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1)) = w(x_0, x_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) +$
 129 $f_X(x_1)) = -w(x_1, x_2)$.

130 Next assume that $F_X(\hat{x}_i) - F_{X'}(\hat{x}_i) = w(x_{i-1}, x_i)$. Then $F_X(x_i) - F_{X'}(x_i) = F_X(\hat{x}_i) + f_X(x_i) -$
 131 $(w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = w(x_{i-1}, x_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) +$
 132 $f_X(x_i)) = -w(x_i, x_{i+1})$.

133 As before we have that for all $x_i < j < x_{i+1}$, we have $F_X(j) - F_{X'}(j) = F_X(j) - F_{X'}(\hat{x}_{i+1}) \leq$
 134 $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})$. Then $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1}) = (F_X(x_i) + P(x_i < x < x_{i+1})) -$
 135 $F_{X'}(x_i) = -w(x_i, x_{i+1}) + 2w(x_i, x_{i+1}) = w(x_i, x_{i+1})$.

136 Finally for $x_m \leq j \leq x_{m+1}$ we have that $F_{X'}(x_m) = 1$ therefore $F_X(x_m) - F_{X'}(x_m) = (1 -$
137 $P(x_m < X < x_{m+1})) - 1 = P(x_m < X < x_{m+1}) = w(x_m, x_{m+1})$, and for every $x_m < j <$
138 x_{m+1} we have $F_X(j) - F_{X'}(j) < (1 - P(x_m < X < x_{m+1})) - 1 < -P(x_m < X < x_{m+1}) =$
139 $-w(x_m, x_{m+1})$ as required. \square

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

141 **Corollary 9.** $d_k(X, X') = \varepsilon(X, S)$.

142 3.2 Step 2: Finding an S that minimizes $\varepsilon(X, S)$

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

147 The KolmogorovApprox algorithm (Algorithm 1) starts by constructing a directed weighted graph
148 G similar to the method of Chakravarty, Orlin, and Rothblum [2]. The nodes V consist of the
149 support of X together with an extra two nodes, $-\infty$ and ∞ for technical reasons, whereas the
150 edges E connect every pair of nodes in one direction (lines 1-2). The weight w of each edge
151 $e = (x, y) \in E$ is determined by one of two cases as in Definition 3. The first is where nodes
152 x or y are the source or target nodes respectively. In this case, the weight is the probability of X
153 to get a value between x and y , non inclusive, i.e., $w(e) = Pr(x < X < y)$. The second case
154 is where x and y are not a source or target nodes, here the weight is the probability of X to get a
155 value between x and y , non inclusive, divided by two i.e., $w(e) = Pr(x < X < y)/2$. The values
156 taken are non inclusive, since we are interested only in the error value. The source node of the
157 shortest path problem at hand corresponds to the $-\infty$ node added to G in the construction phase,
158 and the target node is the extra node ∞ . The set of all solution paths in G , i.e., those starting at
159 $-\infty$ and ending in ∞ with at most m edges, is called $paths(G, -\infty, \infty)$. The goal is to find the
160 path l in $paths(G, -\infty, \infty)$ with the lightest bottleneck (line 3). This can be achieved by using the
161 *Bellman – Ford* algorithm with two tweaks. The first is to iterate the graph G in order to find only
162 paths with length of at most m edges. The second is to find the lightest bottleneck as opposed to
163 the traditional objective of finding the shortest path. This is performed by modifying the manner of
164 “relaxation” to $bottleneck(x) = \min[\max(bottleneck(v), w(e))]$, done also in [14]. Consequently,
165 we find the lightest maximal edge in a path of length $\leq m$, which represents the minimal error,
166 $\varepsilon(X, S)$, defined in Definition ?? where the nodes in path l represent the elements in set S . The
167 approximated random variable X' is then derived from the resulting path l (lines 4-5). Every node
168 $x \in l$ represent a value in the new calculated random variable X' , we than iterate the path l to find
169 the probability of the event $f_{X'}(x)$ as described in Definition 6. For every edge $(x_i, x_j) \in l$ we
170 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
171 $w(x_i, x_j)$ 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
172 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
173 own original weight in X and the probability that already accumulated. If (x_i, x_j) is the last edge
174 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
175 accumulated such that $f_{X'}(x_j) = f_{X'}(x_i) + w(x_i, x_j)$.

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

177 *Proof.* If we consider the vertexes $S = l \setminus \{-\infty, \infty\}$ for a path $l \in paths(G, -\infty, \infty)$ we have
178 that $\max\{w(e) : e \in l\} = \varepsilon(X, S)$. Therefore, line 3 of the algorithm essentially computes a set

Algorithm 1: KolmogorovApprox(X, m)

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

179 $S \in \text{argmin}_{S \subseteq \text{support}(X), |S| \leq m} \varepsilon(X, S)$. By Corollary 9, the variable X' constructed in lines 4 and
180 5 satisfies $d_K(X, X') = \varepsilon(X, S)$ and by the minimality of S and by Proposition 5, it is an optimal
181 approximation. \square

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

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

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

190 Since G is DAG (directed acyclic graph) finding shortest path takes $O(E + V)$. We only need
191 to find paths of length $\leq m$, which takes $O(m(E + V))$. Deriving the new random variable X'
192 from the computed path l takes $O(mn)$. For every node in l (at most m nodes), calculating the
193 probability $P(s < X < \infty)$ takes at most n steps. To conclude, the worst case run-time complexity
194 is $O(n^2 + mn^2 + mn) = O(mn^2)$ and memory complexity is $O(E + V) = O(n^2)$. \square

195 4 A case study and experimental results

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

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

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

214 In the first experiment we focus on is the problem of task trees with deadlines, and consider three
 215 types of task trees. The first type includes logistic problems of transporting packages by trucks and
 216 airplanes (from IPC2 <http://ipc.icaps-conference.org/>). Hierarchical plans of those logistic problems
 217 were generated by the JSHOP2 planner [11] (see example problem, Figure 1, one parallel node with
 218 all descendant task nodes being in sequence). The second type consists of task trees used as execution
 219 plans for the ROBIL team entry in the DARPA robotics challenge (DRC simulation phase), and the
 220 third type is of linear plans (sequential task trees). The primitive tasks in all the trees are modeled as
 221 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 .

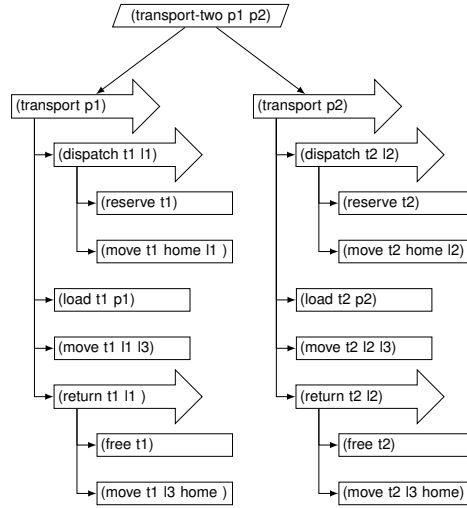


Figure 1: A plan generated by the JSHOP2 algorithm. Arrow shapes represent sequence nodes, parallelograms represent parallel nodes, and rectangles represent primitive nodes.

222

223 We implemented the approximation algorithm for solving the deadline problem with four different
 224 methods of approximation. The first two are for achieving a one-sided Kolmogorov approximation –
 225 the OptTrim [3] and the Trim [4] operators, and the third is a simple sampling scheme. We used those
 226 methods as a comparison to the Kolmogorov approximation with the suggested KolmogorovApprox
 227 algorithm. The parameter m of OptTrim and KolmogorovApprox corresponds to the inverse of ε
 228 given to the Trim operator. Note that in order to obtain some error ε , one must take into consideration
 229 the size of the task tree N , therefore, $m/N = 1/(\varepsilon \cdot N)$. We ran also an exact computation as
 230 a reference to the approximated one in order to calculate the error. The experiments conducted
 231 with the following operators and their parameters: KolmogorovApprox operator with $m = 10 \cdot N$,
 232 the OptTrim operator with $m = 10 \cdot N$, the Trim as operator with $\varepsilon = 0.1/N$, and two simple
 233 simulations, with a different samples number $s = 10^4$ and $s = 10^6$.

234 Table 1 shows the results of the main experiment. The quality of the solutions provided by using the
 235 OptTrim operator are better (lower errors) than those provided by the Trim operator, following the
 236 optimality guarantees, but is interesting to see that the quality gaps happen in practice in each of the
 237 examined task trees. However, in some of the task trees the sampling method produced better results
 238 than the approximation algorithm with OptTrim. Nevertheless, the approximation algorithm comes

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.

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 ?? 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.

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.