Efficient Optimal Approximation of Discrete Random Variables for Estimation of Probabilities of Missing Deadlines

Liat Cohen, Gera Weiss
Department of Computer Science
Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel
{liati,geraw}@cs.bgu.ac.il

Abstract

We present an efficient algorithm that, given a discrete random variable X and a number m, computes a random variable whose support is of size at most m and whose Kolmogorov distance from X is minimal. We present some variants of the algorithm, analyse their correctness and computational complexity, and present a detailed empirical evaluation that shows how they performs in practice. The main application that we examine, which is our motivation for this work, is estimation of the probability of missing deadlines in series-parallel schedules. Since exact computation of these probabilities is NP-hard, we propose to use the algorithms described in this paper to obtain an approximation.

1 Introduction

Various approaches for approximation of probability distributions are studied in the literature (Pettitt and Stephens 1977; Miller and Rice 1983; Vidyasagar 2012; Cohen, Shimony, and Weiss 2015; Pavlikov and Uryasev 2016; Cohen, Grinshpoun, and Weiss 2018). These approaches vary in the types random variables considered, how they are represented, and in the criteria used for evaluation of the quality of the approximations. In this paper we propose an approach for compressing the probability mass function of a random variable X such that the errors added to queries such as $Pr(X \leq t)$, for any $t > 0$, is minimal. In other words, we minimise the Kolmogorov distance between the approximation and the original variable, see alternative definition in Equation (1).

Our main motivation for this work is estimation of the probability for missing deadlines, as described, e.g., in Cohen et al. (Cohen, Shimony, and Weiss 2015; Cohen, Grinshpoun, and Weiss 2018) and in (Kashef and Moayeri 2018). Specifically, when X represents the probability distribution of the time to complete some complex schedule and we cannot afford to maintain the full table of its probability mass function, we propose an algorithm for producing a smaller table, whose size can be specified, such that probabilities for missing deadlines are preserved as much as possible.

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 size is not above a prescribed threshold, where the measures of the quality of the approximation and of its size are as specified in the following two paragraphs.

We measure the quality of an approximation scheme by the distance between random variables and their approximations. Specifically, we use the Kolmogorov distance which is commonly used for comparing random variables in statistical practice and literature. Given two random variables X and X' whose cumulative distribution functions (cdf) 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., (Gibbons and Chakraborti 2011)). We say that X' is a good approximation of X if $d_K(X, X')$ is small. This distance is the basis for the often used Kolmogorov-Smirnoff test for comparing a sample to a distribution or two samples to each other.

The size of a random variable is measured by the size of its support, the set of possible outcomes, $|X| = |\{x\colon Pr(X{=}x) \neq 0\}|$. When probability mass functions are maintained as tables, as done in many implementations of statistical software, the support size is proportional to the memory needed to store the variable and to the complexity of the computations that manipulate it. 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 $|X'| \leq m$ and there is no random variable X'' such that $|X''| \leq m$ and $d_K(X, X'') < d_K(X, X')$.*

In these terms, the main contribution of the paper is an efficient (linear time and constant memory) algorithm that takes X and m as parameters and constructs an optimal m-approximation of X.

The rest of the paper is organised 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, analyse its properties, and prove the main theorems. In Section 4 we demonstrate how the proposed approach performs on the problem of estimating the probability of missing deadlines in series-parallel schedules on randomly generated random variables and compare it to alternative approximation approaches from the literature. The paper is concluded with a discussion and with ideas for future work in Section 5.

2 Related work

The most relevant work related to this paper is the papers on approximations of random variables in the context of estimating deadlines (Cohen, Shimony, and Weiss 2015; Cohen, Grinshpoun, and Weiss 2018). In these papers, X' is defined to be a good approximation of X if $F_{X'}(t) > F_X(t)$ for any t and $\sup_t F_{X'}(t) - F_X(t)$ is small. Note that this measure is not a proper distance measure because it is not symmetric. The motivation given in these papers for using this type of approximation is for cases where overestimation of the probability of missing a deadline is acceptable but underestimation is not. We consider in this paper the same case-studies examined by Cohen et al. and show how the algorithm proposed in this paper performs relative to the algorithms proposed there when both over- and under- estimations are allowed. As expected, the Kolmogorov distance between the approximated and the original random variable is considerably smaller when using the algorithm proposed in this paper.

In the technical level, the problem we study in this paper is similar to the problem of approximating a set of 2-D points by a step function. The study of this problem was motivated by query optimisation and histogram constructions in database management systems (Guha 2008; Guha and Shim 2007; Guha, Shim, and Woo 2004; Jagadish et al. 1998; Karras, Sacharidis, and Mamoulis 2007; Fournier and Vigneron 2011) and computational geometry (Díaz-Bánez and Mesa 2001; Fournier and Vigneron 2008). There are, however, two technically significant differences between the problem studied in the context of databases and the problem we analyse in this paper. The first difference is that in the context of approximation of random variables, the step function (which is the cumulative distribution function in our context) must end with a value one, since we are dealing with random variables which sums to one. The second difference is that the first step is not counted because there is no need to put a value in the support of the approximated random variable to generate this first step. These cannot be addressed by adding a constant (two) to m because the first step is always present and because the requirement to end with the value one, restricts the set of eligible step functions.

Another relevant prior work is the theory of Sparse Approximation (aka Sparse Representation) that deals with sparse solutions for systems of linear equations, as follows. Given a matrix $D \in \mathbb{R}^{n\times p}$ and a vector $x \in \mathbb{R}^n$, the most studied sparse representation problem is finding

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

where $\|\alpha\|_0 = |\{i \in [p] : \alpha_i \neq 0\}|$ is the ℓ_0 pseudo-norm, counting the number of non-zero coordinates of α. This problem is known to be NP-hard with a reduction to NP-complete subset selection problems. In these terms, using also the ℓ_∞ norm that represents the maximal coordinate and the ℓ_1 norm that represents the sum of the coordinates, our problem can be phrased as:

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

where D is the lower unitriangular matrix, x is related to X such that the ith coordinate of x is $F_X(x_i)$ where $\text{support}(X) = \{x_1 < \cdots < x_n\}$ and α is related to X' such that the ith coordinate of α is $f_{X'}(x_i)$. The functions F_X and $f_{X'}$ represent, respectively, the cumulative distribution function of X and the mass distribution function of X', i.e., the coordinates of x are positive and monotonically increasing and its last coordinate is one.

The presented work is also related to the research on binning in statistical inference. Consider, for example, the problem of credit scoring (Zeng 2017) that deals with separating good applicants from bad applicants where the Kolmogorov–Smirnov statistic KS is a standard measure. The KS comparison is often preceded by a procedure called binning where small values in the probability mass function are moved to nearby values. There are many methods for binning (Mays 2001; Refaat 2011; Bolton and others 2010; Siddiqi 2012). In this context, our algorithm can be considered as a binning strategy that provides optimality guarantees with respect to the Kolmogorov distance.

Our study is also related to the work of Pavlikov and Uryasev (2016), where a procedure for producing a random variable X' that optimally approximates a random variable X is presented. Their approximation scheme, achieved using linear programming, is designed for a different notion of distance called CVaR. The contribution of the present work in this context is that our method is direct, not using linear programming, thus allowing tighter analysis of time and memory complexities. Also, our method is designed for minimising the Kolmogorov distance that is more prevalent in applications. For comparison, in Section 4 we briefly discuss the performance of linear programming approach similar to the one proposed in (Pavlikov and Uryasev 2016) for the Kolmogorov distance and compare it our algorithm.

A problem very similar to ours is termed "order reduction" by Vidyasagar in (Vidyasagar 2012). There, the author defines an information-theoretic based distance between discrete random variables and studies the problem of finding a variable whose support is of size m and its distance from X is as small as possible (where X and m are given). The main difference between this and the problem studied in this paper, is that Vidyasagar examines a different notion of distance. Vidyasagar proves that computing the distance (that he considers) between two probability distributions, and computing the optimal reduced order approximation, are both NP-hard problems, because they can both be reduced to nonstandard bin-packing problems. He then develops efficient greedy approximation algorithms. In contrast, our study shows that there are efficient solutions to these problems when the Kolmogorov distance is considered.

3 Algorithms for optimal approximation

We begin with presenting an algorithm for solving a problem that is dual to the m-approximation problem: given a random variable X and $0 \leq \varepsilon \leq 1$, find a new random variable X' such that $d_k(X, X') \leq \varepsilon$ and $|X'|$ is minimal.

We assume that the input to Algorithm 1 that solves the dual problem is a representation of the variable X as a sorted CDF, i.e., as an array $X_{CDF} = \{(x_i, c_i)\}_{i=1}^n$ such that $c_i = Pr(X \leq x_i)$ and $\text{support}(X) = \{x_1 < \cdots < x_n\}$. In line 2 we perform a single pass over X_{CDF} starting from the first

```
Algorithm 1: $dual(\{(x_i, c_i)\}_{i=1}^{n}, \varepsilon)$
1  $f \leftarrow 1, l \leftarrow n{+}1$
2  while $c_f \leq \varepsilon$ do $f \leftarrow f + 1$
3  while $c_{l-1} \geq 1 - \varepsilon$ do $l \leftarrow l - 1$
4  $S \leftarrow \emptyset$, $s \leftarrow 0$, $b \leftarrow f$, $e \leftarrow f$
5  while $e < l$ do
6      while $c_{e+1} - c_b \leq 2\varepsilon \wedge e < l$ do
7          $e \leftarrow e + 1$
8      $S \leftarrow S \cup \{(x_b, (c_b + c_e)/2 - s)\}$
9      $s \leftarrow (c_b + c_e)/2$, $b \leftarrow e \leftarrow e + 1$
10 $S \leftarrow S \cup \{(x_l, 1 - s)\}$
11 return A r.v. $X'$ such that $Pr(X'{=}x) = c$ if there is $c$
       such that $(x, c) \in S$ and $Pr(X'{=}x) = 0$ otherwise.
```

index to find the last index where the cumulative distribution function of X is less than ε, this index is stored in the variable f. In line 3 we start from the last index of X_{CDF} which is n, and seek in decreasing order for the last index where the cumulative distribution function of X is less than $1 - \varepsilon$. This index is stored at the variable l. The last part of the algorithm is a pass on all elements of X_{CDF} between the indices f and l in order to construct the random variable X'. In lines 6-7 we check if the difference between the first element in the current set of indices (which we call b) and the last element in the current set of indices (which we call $e + 1$) is less than 2ε. If yes, we add the pair $(x_b, (c_b + c_e)/2 - s)$ to the set S. Otherwise, we add one to e. See running Example 2.

Example 2. *When dual is invoked with the parameters* $X{=}\{(1, 0.3), (2, 0.7), (3, 0.9), (4, 1)\}$ *and* $\varepsilon{=}0.1$*: Lines 2 and 3 set* $f = 1$ *and* $l = 4$*. After an iteration of the main loop (line 5),* $S = \{(1, 0.3)\}$*,* $s = 0.3$*, and* $b = e = 2$*. After a second iteration,* $S = \{(1, 0.3), (2, 0.5)\}$*,* $s = 0.8$*, and* $b = e = 4$*. At the end,* $S = \{(1, 0.3), (2, 0.5), (4, 0.2)\}$*.*

Proposition 3. *If* $\{(x_i, c_i)\}_{i=1}^{n}$ *is such that* $c_i = Pr(X \leq x_i)$ *and* $\text{support}(X) = \{x_1 < \cdots < x_n\}$ *then*

$$dual(\{(x_i, c_i)\}_{i=1}^{n}, \varepsilon) \in \underset{X' \in \bar{\mathcal{B}}(X,\varepsilon)}{\arg\min} |X'|$$

where $\bar{\mathcal{B}}(X, \varepsilon) = \{X' \colon d_K(X, X') \leq \varepsilon\}$*.*

Proof. The number of level sets of the CDF of X' is minimal: (1) The first and the last sets are of maximal length; (2) By construction, an extension of any of the other sets to the right will generate a random variable whose Kolmogorov distance from X is bigger than ε; and (3) Extension of a level set to the left may either leave the number of level sets unchanged or combine it with the previous set, which will enlarge the Kolmogorov distance because it is equivalent to extending the previous set to the right. Thus, there is no random variable whose Kolmogorov distance from X is smaller or equal to ε and its support is smaller than $|X'|$. $\square$

Proposition 4. $dual(\{(x_i, c_i)\}_{i=1}^{n}, \varepsilon)$ *runs in time* $O(n)$*, using* $O(n)$ *memory.*

Proof. This algorithm describes a single pass over $\{(x_i, c_i)\}_{i=1}^{n}$. Lines 2 and 3 are easy to follow, each takes $O(n)$ in the worst case. Lines 5-9 also describe a single pass since the counter e is updated to $e + 1$ at most n times. Altogether we get run-time complexity of $O(n)$. We are constructing the set S which is of size n in the worst case, therefore, memory complexity is $O(n)$. $\square$

The first solution to the m-approximation problem we present is the $binsApprox(X, m)$ algorithm which is based on (Díaz-Bánez and Mesa 2001). There are couple of significant changes between our $binsApprox(X, m)$ algorithm and the algorithm suggested by Díaz-Bánez and Mesa (2001) addressing the differences presented in the Related work section. The algorithm $binsApprox(X, m)$ gets as input a random variable X and some number m. Again, we do not mind the original representation of X since we can transform it to a sorted CDF representation in $O(nlog(n))$ run-time as in line 1 which we call X_{CDF}. In lines 3-8 we compute all possible errors, in other words, all possible $d_K(X, X')$ such that X' is an approximation of X and $\text{support}(X') \subseteq \text{support}(X)$. The error is just the difference between the CDF values of every two elements in X_{CDF}. After computing the set E, we sort it (line 9) in order to perform a binary search. In lines 10-19 we perform a binary search over the set E, in every step of the binary search we run the $X' = dual(X, \varepsilon)$ algorithm. If the size of $|X'| > m$ then we know that the error ε is too small and need to extend the search in the right side of E, if the size of $|X'| < m$ then we know that the error ε is too big and need to extend the search in the left side of E, otherwise we found the correct m. We run the search twice to handle the extreme case where $m \notin \{|dual(X, \varepsilon)| \colon \varepsilon \in E\}$. In this case we may end up with a variable that is not optimal because it can be improved by using m'' such that $m < m'' < m'$. We find the optimal m'' by the second round ($i = 2$) that run after line 18 that sets m to be the m' found in the first round.

```
Algorithm 2: $binsApprox(X, m)$
1  Let $\{(x_i, c_i)\}_{i=1}^{n}$ be such that $c_i = Pr(X \leq x_i)$ and
     $\text{support}(X) = \{x_1 < \cdots < x_n\}$.
2  $E \leftarrow \emptyset$
3  for $i \leftarrow 1$ to $n - 1$ do
4      for $j \leftarrow i + 1$ to $n - 1$ do
5          if $i = 1$ then
6              $E \leftarrow E \cup \{c_j\}$
7          $E \leftarrow E \cup \{(c_j - c_i)/2\}$
8      $E \leftarrow E \cup \{1 - c_i\}$
9  Let $e_1 < \cdots < e_{n'}$ be such that $E = \{e_1, \ldots, e_{n'}\}$
10 for $i \leftarrow 1$ to $2$ do
11     $l \leftarrow 1, r \leftarrow n', k \leftarrow 1, k' \leftarrow 0$
12     while $k \neq k'$ do
13         $k' \leftarrow k, k \leftarrow \lceil (l + r)/2 \rceil$
14         $m' \leftarrow |dual(X, e_k)|$
15         if $m' < m$ then $l \leftarrow k$
16         if $m' > m$ then $r \leftarrow k - 1$
17         if $m' = m$ then $r \leftarrow k$
18     $m \leftarrow m'$
19 return $dual(X, e_k)$
```

Proposition 5. $binsApprox(X, m) \in \arg\min_{|X'| \leq m} d_K(X, X')$

Proof. Since lines 10-18 is a binary search of the smallest $e_k \in E$ such that $|dual(X, e_k)| \leq m$, we only need to prove that $E = E' = \{d_K(X, dual(X, e_m)) \colon m = 1, \ldots, n\}$. To see that this is true, note that every element in E' corresponds to a distance of a level set of the CDF of $dual(X, m)$ from the CDF of X (because we seek a variable whose support is included in the original one). Line 6 of the algorithm adds all the distances from level sets of height zero, Line 8 adds the distances from level sets of height one, and Line 7 adds all the distances from all other possible level sets. Note that the distance is monotonic with the support size. □

Proposition 6. *The $binsApprox(X, m)$ algorithm runs in time $O(n^2 log(n))$, using $O(n^2)$ memory where $n = |X|$.*

Proof. In the first part of the algorithm, lines 2-8, we construct the set E which takes $O(n^2)$ run-time. In the second part of the algorithm, line 9, we sort the set E which takes $O(n^2 log(n^2)) = O(n^2 log(n))$ run-time. The third part of the algorithm, lines 10-19, describes a binary sort over the set E where in each step of the sorting we run the $dual$ algorithm, which takes $O(nlog(n^2)) = O(nlog(n))$ run-time. We run this part twice then we get $O(2nlog(n))$. All together, the run-time complexity is $O(n^2 + n^2 log(n) + 2nlog(n)) = O(n^2 log(n))$ and the memory complexity is for storing the set E which is $O(n^2)$. □

Towards an improved algorithm let us introduce the matrix $E = (e_{i,j})_{i,j=1}^{\infty}$ defined by:

$$e_{i,j} = \begin{cases} 1 - c_{n+1-j} & \text{if } j \leq n \wedge i = n; \\ c_i & \text{if } i \leq n \wedge j = n+1; \\ (c_i - c_{n+1-j})/2 & \text{if } i < n \wedge j \leq n \wedge i{+}j \geq n; \\ 0 & \text{if } i{+}j < n; \\ 1 & \text{otherwise.} \end{cases}$$

Where $c_1, \ldots, c_n$ are as in the first line of Algorithm 2. It is easy to see that the set of values in this matrix are the elements of the set E in Algorithm 2. An additional useful fact is that E is a sorted matrix:

Lemma 7. *If $i \leq i'$ and $j \leq j'$ then $e_{i,j} \leq e_{i',j'}$.*

Proof. Since the c_is are monotonically increasing, $\forall i, j, 0 \leq c_i - c_{n+1-j} \leq c_i - c_{n+1-(j+1)}$ and $\forall i, j, 0 \leq c_i - c_{n+1-j} \leq c_{i+1} - c_{n+1-j}$, moreover, 0 is the minimal value of E. The elements in the last row and the last column in E are keeping the terms of sorted matrix. It suffices to compare row n with row $n-1$ and show that $1 - c_{n+1-j} \geq (c_{n-1} - c_{n+1-j})/2$. After some manipulation we get that $2 - c_{n+1-j} \geq c_{n-1}$ which is true because $0 \leq c \leq 1$. Next, it is suffice to compare column n with only column $n+1$ and show that $c_i \geq (c_i - c_{n+1-n})/2$. After some manipulation we get that $2c_i \geq c_i - c_1$ which is true because $0 \leq c_1 \leq c_i \leq 1$. □

The fact that this matrix is sorted allows us to use the saddleback search algorithm listed as Algorithm 3. The algorithm starts at the top right entry of the matrix $(e_{i,j})_{i=1..n, j=1..(n+1)}$ and traverses it as follows. If it hits an entry e such that

Algorithm 3: $sdlbkApprox(X, m)$

```
1 Let $\{(x_i, c_i)\}_{i=1}^n$ be such that $c_i = Pr(X \leq x_i)$ and
    support$(X) = \{x_1 < \cdots < x_n\}$.
2 $i \leftarrow 1, j \leftarrow n{+}1, S \leftarrow \emptyset$
3 while $i < n \wedge j \geq 1$ do
4     $m' \leftarrow |dual(X, e_{i,j})|$
5     if $m' \leq m$ then $j \leftarrow j - 1$
6     if $m' > m$ then $i \leftarrow i + 1$
7     if $m' \geq m$ then $S \leftarrow S \cup (m', e_{i,j})$
8 $e \leftarrow \min\{e \colon (m', e) \in \arg\max_{(m',e) \in S} m'\}$
9 return $dual(X, e)$
```

$|dual(X, e)| \leq m$ it goes left, otherwise it goes down. This assures that the minimal e such that $|dual(X, e)| \leq m$ is visited after at most $n + 1$ steps. The optimal random variable is found by brute-force search over the visited entries.

Proposition 8. $sdlbkApprox(X, m) \in \arg\min_{|X'| \leq m} d_K(X, X')$

Proof. The algorithm traverses all the frontier between those es that satisfy $|dual(X, e)| \leq m$ and those that do not satisfy it. Since all the entries that satisfy the condition are recorded in S and considered in the brute-force phase in line 7, the minimal satisfying e is found. □

Proposition 9. *The $sdlbkApprox(X, m)$ algorithm runs in time $O(n^2)$, using $O(n)$ memory where $n = |X|$.*

Proof. At each step of the loop in lines 3-6, either j decreases or i increases, thus the loop can be executed at most $2n$ times. Since we execute $dual$ once in a loop round, the total time complexity is $O(n^2)$. Storing visited states in S on line 6 requires $O(n)$ memory. □

The problem with the saddleback algorithm is that it needs to run $dual$ at every step so it has a quadratic time complexity. Since we cannot find the required entry of the matrix in less than n steps, we can only reduce the complexity by proposing an algorithm that does not execute $dual$ in all the steps, only in $\log(n)$ of them. Such an algorithm, based on Section 2.1 of (Fournier and Vigneron 2011), is listed as Algorithm 4. The algorithm maintains a set S of sub-matrices of $(e_{i,j})_{i=1..2^{\lceil \log_2(n) \rceil}, j=1..2^{\lceil \log_2(n+1) \rceil}}$. At each round of its execution, each sub-matrix is split to four and then about three quarters of the matrix are discarded. At the end, at most four scalar matrices remain containing the index of the entry we seek. Note that this algorithm runs on a matrix that, in the worst case, can be almost four times bigger than the matrix traversed by the saddleback algorithm. This, of course, does not affect the asymptotic complexity, but it may matter when dealing with relatively small random variables.

Theorem 10. $linApprox(X, m) \in \arg\min_{|X'| \leq m} d_K(X, X')$

Proof. In line 10 we only discard sub-matrices whose minimal entry (at the top left) is larger than an entry that we prefer. In line 12 we only discard sub-matrices whose maximal entry (at

Algorithm 4: $linApprox(X, m)$

```
1  Let $\{(x_i, c_i)\}_{i=1}^n$ be such that $c_i = Pr(X \leq x_i)$ and
   $\text{support}(X) = \{x_1 < \cdots < x_n\}$.
2  $S \leftarrow \{((1,1), (2^{\lceil \log_2(n) \rceil}, 2^{\lceil \log_2(n+1) \rceil}))\}$, $S' \leftarrow \emptyset$
3  while $S \neq S'$ do
4      $S' \leftarrow S$, $S \leftarrow \bigcup_{s \in S} split(s)$
5      $e^- \leftarrow median(\{e_{i_1,j_1} : ((i_1,j_1),(i_2,j_2)) \in S\})$
6      $e^+ \leftarrow median(\{e_{i_2,j_2} : ((i_1,j_1),(i_2,j_2)) \in S\})$
7      for $e \in \{e^-, e^+\}$ do
8          $m' \leftarrow |dual(X, e)|$
9          if $m' \leq m$ then
10             $S \leftarrow$
               $S \setminus \{((i_1,j_1),(i_2,j_2)) \in S : e_{i_1,j_1} > e\}$
11         else
12             $S \leftarrow$
               $S \setminus \{((i_1,j_1),(i_2,j_2)) \in S : e_{i_2,j_2} \leq e\}$
13 $S'' \leftarrow \{(|dual(X, e_{i,j})|, e_{i,j}) : ((i,j),(i,j)) \in S\}$
14 $e \leftarrow \min\{e : (m', e) \in \arg\max_{(m',e) \in S''} m'\}$
15 return $dual(X, e)$
```

Function $split(((i_1, j_1), (i_2, j_2)))$

```
1  $j^- \leftarrow \lfloor (j_1 + j_2)/2 \rfloor$, $j^+ \leftarrow \lceil (j_1 + j_2)/2 \rceil$
2  $i^- \leftarrow \lfloor (i_1 + i_2)/2 \rfloor$, $i^+ \leftarrow \lceil (i_1 + i_2)/2 \rceil$
3  return
       $\{((i_1,j_1),(i^-,j^-)), ((i_1,j^+),(i^-,j_2)),$
       $((i^+,j_1),(i_2,j^-)), ((i^+,j^+),(i_2,j_2))\}$
```

the bottom right) is smaller or equal to an entry that does not meet our condition. □

Theorem 11. *The* $linApprox(X, m)$ *algorithm runs in time* $O(n \log(n))$*, using* $O(1)$ *memory where* $n = |X|$.

Proof. The dimension of each matrix is halved in each round, thus the loop is executed $O(\log(n))$ rounds. Since $dual$ is called one in each round and a finite number of times at the end, the time complexity is $O(n \log(n))$. □

4 Experimental evaluation

We describe below several experiments that show how linApprox performs in practice in different applications and domains. All algorithms were implemented in Python and the experiments were executed on a hardware comprised of an Intel i5-6500 CPU @ 3.20GHz processor and 8GB memory. The algorithms of Cohen et al. were taken "as is" from in the supplementary material to (Cohen, Shimony, and Weiss 2015).

Repetitive support size minimization. One use of support size minimization is when commutations that involve summations of random variables slow due to an exponential growth in the support of convolutions of random variables (Cohen, Shimony, and Weiss 2015). A key action in coping with this situation is reduction of the support size by replacing the summed random variable by an approximation of it that has a smaller support size. Previous work such as the work of Cohen et al. in (2015; 2018) handle this reduction using weaker or sub-optimal notion of approximation than ours.

As we proved above, given m, a single step of linApprox guarantees an optimal m-approximation. However in the setting considered here we need to repetitively use linApprox, thus the optimality of the eventually obtained random variable is not guaranteed. In light of this, we tested the accuracy of the repetitive-linApprox to see how it performs against the tools of (Cohen, Shimony, and Weiss 2015; Cohen, Grinshpoun, and Weiss 2018) using their benchmarks. These benchmarks are taken from the area of task trees with deadlines, a sub area of the well-established hierarchical planning (Dean, Firby, and Miller 1988; Alford et al. 2016; Xiao et al. 2017).

We estimated the probability for meeting deadlines in plans, as described in Cohen et al. (2015; 2018), and experimented with four different methods of approximation. The first two, OptTrim (Cohen, Grinshpoun, and Weiss 2018) and the Trim (Cohen, Shimony, and Weiss 2015), are taken from the repository provided by the authors and are designed for achieving only a one-sided Kolmogorov approximation - a weaker notion of approximation than the Kolmogorov approximation analyzed in this work. The third method is a simple sampling scheme and the fourth is our Kolmogorov approximation obtained by the proposed linApprox algorithm. The parameters for the different methods were chosen in a compatible way, M is the maximal support size, N is the number of nodes of the plan network and s is the number of samples. We ran also an exact computation as a reference to the approximated one in order to calculate the errors.

Task Tree	M	linApprox	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
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 experiment. The quality of the solutions obtained with the linApprox operator are better than those obtained by the Trim and OptTrim operators as expected. In some of the task trees, the sampling method produced better results than linApprox. Still, the linApprox approximation algorithm comes with an inherent advantage of providing exact quality guarantees, as opposed to sampling where the best one can hope for is probabilistic guarantees.

Run-time comparison. Similarly to accuracy and error computation, we also conducted empirical evaluation to examine the run-time in practice of the discussed algorithms. Figure 1 presents a comparison of the run-time performances of an exact computation and approximated computations with

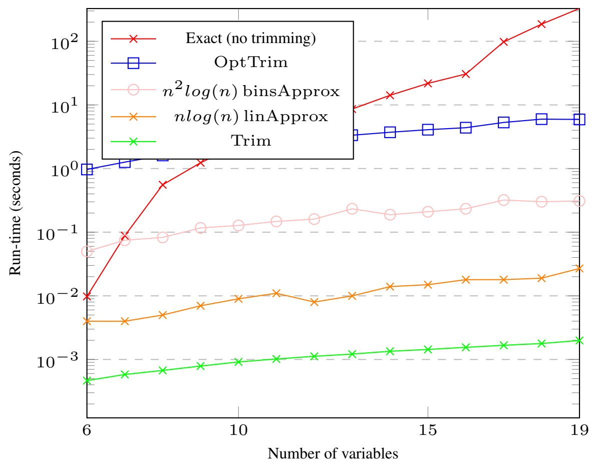

Figure 1: Run-time of a long computation with $\mathrm{linApprox}$, $\mathrm{binsApprox}$, $\mathrm{OptTrim}$, Trim, and without any trimming.

$\mathrm{linApprox}$, $\mathrm{binsApprox}$, $\mathrm{OptTrim}$ and Trim as operators. We examined four versions of approximation each of which with different run-time but only $\mathrm{linApprox}$ and $\mathrm{binsApprox}$ produce optimal Kolmogorov approximation. The computation is a summation of a sequence of random variables with support size of m=10, where the number n of variables varies from 6 to 19. In this experiment, we executed the approximation algorithm with m=10 after performing each convolution between two random variables, in order to maintain a support size of 10 in all intermediate computations. Equivalently, we executed the Trim operator with $\varepsilon = 0.1$. The results clearly show the exponential run-time of the exact computation, caused by the convolution between two consecutive random variables. In fact, in the experiment with N=20, the exact computation ran out of memory. These results illuminate the advantage of the proposed $\mathrm{linApprox}$ algorithm that balances between solution quality and run-time performance – while there exist other, faster, methods (e.g., Trim), $\mathrm{linApprox}$ provides high-quality solutions in $O(nlog(n))$ time, which is especially important when an exact computation is not feasible, due to time or memory. In general, Figure 1 is consistent with the theory and show a good fit to the complexity analysis.

Single step support minimisation. To better understand the quality gaps in practice between $\mathrm{linApprox}$, $\mathrm{OptTrim}$, and Trim, we tested their performance on random variables with n=100, and different ms. Note that the error obtained by $\mathrm{linApprox}$ is optimal while the other methods are not optimised for the Kolmogorv distance. In each instance of the experiment, a random variable is randomly generated by choosing the probabilities of each element in the support uniformly and then normalise these probabilities to sum to 1.

Figure 2 presents the error produced by the above methods. The depicted results are averages over fifty instances of random variables. The curves in the figure show the average error of $\mathrm{OptTrim}$ and Trim operators with comparison to the average error of the optimal approximation provided by $\mathrm{linApprox}$ as a function of m. It is evident from this graphs that increasing

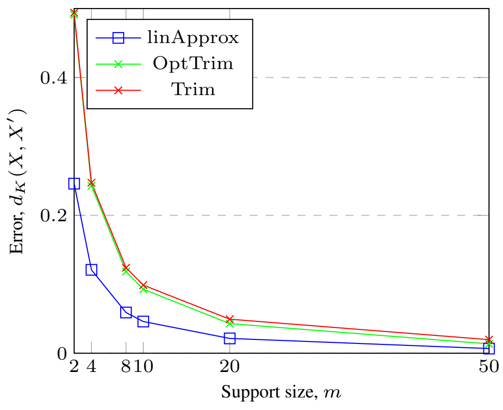

Figure 2: Error comparison between $\mathrm{linApprox}$, $\mathrm{OptTrim}$, and Trim on randomly generated variables as function of m.

the support size of the approximation m reduces the error, as expected, in all three methods. However, the errors produced by the $\mathrm{linApprox}$ are significantly smaller, a half of the error produced by $\mathrm{OptTrim}$ and Trim.

Comparison to Linear Programming. We also compared the run-time of $\mathrm{linApprox}$ with a linear programming (LP) algorithm that guarantees optimality, as described and discussed in (Pavlikov and Uryasev 2016). We used the "Minimize" function of Wolfram Mathematica as a state-of-the-art implementation of linear programming, encoding the problem by the LP problem $\min_{\alpha \in \mathbb{R}^n} \|x - \alpha\|_\infty$ subject to $\|\alpha\|_0 \leq m$ and $\|\alpha\|_1$=1. The run-time comparison results were clear and persuasive: $\mathrm{linApprox}$ significantly outperforms the LP algorithm. For a random variable with support size n=10 and $m = 5$, the LP algorithm run-time was 850 seconds, where the $\mathrm{linApprox}$ algorithm run-time was less than a second. For n=100 and m=5, the $\mathrm{linApprox}$ algorithm run-time was 0.14 seconds and the LP algorithm took more than a day. 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 $\mathrm{linApprox}$.

5 Discussion and future work

We developed an efficient algorithm for computing optimal approximations of random variables where the approximation quality is measured by the Kolmogorov distance. As demonstrated in the experiments, our algorithm improves on the approach of Cohen et al. (2015) and (2018) in that it finds an optimal two sided Kolmogorov approximation, and not just one sided. In addition, the algorithm $\mathrm{linApprox}$ presented in this paper is very efficient with complexity of $O(nlog(n))$ as proved in Theorem 11 and showed in Figure 1. Beyond the Kolmogorov measure studied here, we believe that similar approaches may apply also to total variation, to the Wasserstein distance, and to other measures of approximations.

Acknowledgments. This research was supported by the Lynn and William Frankel Centre for Computer Science at Ben-Gurion University and by the Israel Science Foundation.

References

Alford, R.; Shivashankar, V.; Roberts, M.; Frank, J.; and Aha, D. W. 2016. Hierarchical planning: Relating task and goal decomposition with task sharing. In *IJCAI*, 3022–3029.

Bolton, C., et al. 2010. *Logistic regression and its application in credit scoring*. Ph.D. Dissertation, University of Pretoria.

Cohen, L.; Grinshpoun, T.; and Weiss, G. 2018. Optimal approximation of random variables for estimating the probability of meeting a plan deadline. In *AAAI*, 6327–6334.

Cohen, L.; Shimony, S. E.; and Weiss, G. 2015. Estimating the probability of meeting a deadline in hierarchical plans. In *IJCAI*, 1551–1557.

Dean, T.; Firby, R. J.; and Miller, D. 1988. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence* 4(3):381–398.

Díaz-Báñez, J. M., and Mesa, J. A. 2001. Fitting rectilinear polygonal curves to a set of points in the plane. *European Journal of Operational Research* 130(1):214–222.

Fournier, H., and Vigneron, A. 2008. Fitting a step function to a point set. In *European Symposium on Algorithms*, 442–453. Springer.

Fournier, H., and Vigneron, A. 2011. Fitting a step function to a point set. *Algorithmica* 60(1):95–109.

Gibbons, J. D., and Chakraborti, S. 2011. Nonparametric statistical inference. In *International encyclopedia of statistical science*. Springer. 977–979.

Guha, S., and Shim, K. 2007. A note on linear time algorithms for maximum error histograms. *IEEE Transactions on Knowledge and Data Engineering* 19(7):993–997.

Guha, S.; Shim, K.; and Woo, J. 2004. Rehist: Relative error histogram construction algorithms. In *VLDB*, 300–311.

Guha, S. 2008. Posterior simulation in the generalized linear mixed model with semiparametric random effects. *J. Comput. Graph. Statist.* 17(2):410–425.

Jagadish, H. V.; Koudas, N.; Muthukrishnan, S.; Poosala, V.; Sevcik, K. C.; and Suel, T. 1998. Optimal histograms with quality guarantees. In *VLDB*, volume 98, 24–27.

Karras, P.; Sacharidis, D.; and Mamoulis, N. 2007. Exploiting duality in summarization with deterministic guarantees. In *SIGKDD*, 380–389.

Kashef, M., and Moayeri, N. 2018. Random-deadline missing probability analysis for wireless communications in industrial environments. In *WFCS*, 1–8.

Mays, E. 2001. *Handbook of credit scoring*. Global Professional Publishi.

Miller, A. C., and Rice, T. R. 1983. Discrete approximations of probability distributions. *Management Science* 29(3):352–362.

Pavlikov, K., and Uryasev, S. 2016. CVaR distance between univariate probability distributions and approximation problems. Technical Report 2015-6, University of Florida.

Pettitt, A. N., and Stephens, M. A. 1977. The kolmogorov-smirnov goodness-of-fit statistic with discrete and grouped data. *Technometrics* 19(2):205–210.

Refaat, M. 2011. *Credit Risk Scorecard: Development and Implementation Using SAS*. Lulu. com.

Siddiqi, N. 2012. *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons.

Vidyasagar, M. 2012. A metric between probability distributions on finite sets of different cardinalities and applications to order reduction. *IEEE Transactions on Automatic Control* 57(10):2464–2477.

Xiao, Z.; Herzig, A.; Perrussel, L.; Wan, H.; and Su, X. 2017. Hierarchical task network planning with task insertion and state constraints. In *IJCAI*.

Zeng, G. 2017. A comparison study of computational methods of kolmogorov–smirnov statistic in credit scoring. *Communications in Statistics-Simulation and Computation* 46(10):7744–7760.