On the Convolution Efficiency for Probabilistic Analysis of Real-Time Systems

Filip Marković
Mälardalen University, Västerås, Sweden

Alessandro Vittorio Papadopoulos
Mälardalen University, Västerås, Sweden

Thomas Nolte
Mälardalen University, Västerås, Sweden

Abstract

This paper addresses two major problems in probabilistic analysis of real-time systems: space and time complexity of convolution of discrete random variables. For years, these two problems have limited the applicability of many methods for the probabilistic analysis of real-time systems, that rely on convolution as the main operation. Convolution in probabilistic analysis leads to a substantial space explosion and therefore space reductions may be necessary to make the problem tractable. However, the reductions lead to pessimism in the obtained probabilistic distributions, affecting the accuracy of the timing analysis. In this paper, we propose an optimal algorithm for down-sampling, which minimises the probabilistic expectation (i.e., the pessimism) in polynomial time.
The second problem relates to the time complexity of the convolution between discrete random variables. It has been shown that quadratic time complexity of a single linear convolution, together with the space explosion of probabilistic analysis, limits its applicability for systems with a large number of tasks, jobs, and other analysed entities. In this paper, we show that the problem can be solved with a complexity of $\mathcal{O}(n\log(n))$, by proposing an algorithm that utilises circular convolution and vector space reductions. Evaluation results show several important improvements with respect to other state-of-the-art techniques.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms; Computer systems organization → Real-time system specification

Keywords and phrases Probabilistic analysis, Random variables, Algorithm Complexity

Digital Object Identifier 10.4230/LIPIcs.ECRTS.2021.16

Supplementary Material *Software (ECRTS 2021 Artifact Evaluation approved artifact)*: `https://doi.org/10.4230/DARTS.7.1.1`

Funding This work was supported by the Swedish Research Council (VR) via the project "Practical Probabilistic Timing Analysis of Real-Time Systems (PARIS)", and by the Knowledge Foundation (KKS) via the project "Federated Choreography of an Integrated Embedded Systems Software Architecture (FIESTA)".
Alessandro Vittorio Papadopoulos: Swedish Research Council (VR) via the project "Pervasive Self-Optimizing Computing Infrastructures (PSI)".

Acknowledgements We want to thank Davor Čirkinagić who borrowed his computing system for performing the evaluation. Also, we are very grateful to the anonymous reviewers for their comments.

1 Introduction

In the last decades, probabilistic timing analysis has emerged as an important concept for assurance of real-time guarantees in various fields [10]. Compared to the deterministic-based worst-case execution time model, a probabilistic model of time parameters offers more expressiveness and a higher degree of representation of the actual system behaviour. Besides that, it can be more applicable for soft real-time systems, and systems with below-worst-case provisioning, that in general are more present than the strictly hard-real time systems.

One of the major concepts that has been used in the analysis of real-time systems is the linear convolution, which is also known as *addition*, when used on discrete random variables. This operation has been studied and adjusted for the analysis of real-time systems in many papers throughout the years [12,19,20,22], while its mere use is present in almost all research areas of probabilistic real-time systems [10].

However, as shown by Davis and Cucu-Grosjean [10], one of the main obstacles for probabilistic timing analysis is the fact that the linear convolution in the worst-case exhibits $\mathcal{O}(n^2)$ time and space complexity. This problem was first addressed with distribution down-sizing techniques [19,20,22,25], and later with the analytical methods [6–8,27,28] for the determination of deadline-miss probability. There are many benefits of using analytical methods, the major one being the reduction of the time and space complexity. However, analytical methods come with some limitations: (i) they often introduce over-approximation in the resulting deadline-miss probabilities [28], and (ii) the most accurate methods rely on convolution, e.g., *Pruning* and *Unify*, proposed by von der Brüggen et al. [28]. Moreover, analytical methods do not provide comprehensive information on the resulting probabilistic distributions. This can be very important for many present problems in the analysis of real-time systems since there are many more potential goals other than determining deadline miss probabilities, e.g., computing cache-miss probability, analysis of random replacement caches [11], analysis of the tasks with multiple probabilistic parameters [18], etc.

This research. There is a rich area of past and future research that is limited by the space and time complexity of linear convolution. In this paper, we focus on further reducing the space and time complexity of convolution-based analyses, as a fundamental operator for the probabilistic analysis of real-time systems. One of the main exploration lines seized in this paper is the concept from mathematics and signal processing known as the *circular convolution* [15,23]. The main contributions of this paper are:

- An algorithm for optimal down-sampling of random variables in terms of probabilistic expectation is proposed, which represents the quantitative degree of pessimism when analysing real-time systems. (Section 3)
- Methods which reduce the time complexity of convolution between two random variables from $\mathcal{O}(n^2)$ (linear convolution from state-of-the-art) to $\mathcal{O}(n\log(n))$. (Section 4)

The results of evaluation show that the proposed methods can be applicable for probabilistic analysis of real-time systems even with large numbers of analysed entities.

Organisation. The remainder of this paper is organised as follows. In Section 2, we describe the basic terminology and mathematical notation used in the paper. Section 3 describes the proposed algorithm for optimal down-sampling of random variables. Section 4 presents algorithms to reduce the time complexity of addition between two random variables. The evaluation is described in Section 5, the related work is presented in Section 6, and the paper is concluded with Section 7.

2 Terminology and mathematical notation

▶ **Definition 1** (Discrete Random Variable). *A discrete random variable X on the probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is defined to be a measurable function $X : \Omega \to \mathbb{R}$ such that the image $X(\Omega)$ is a countable subset of $\mathbb{R}$, and $\{\omega \in \Omega : X(\omega) = x\} \in \mathcal{F}$ for $x \in \mathbb{R}$.*

Table 1 List of important symbols used in the paper.

Symbol	Brief explanation
X, Y, Z	Discrete random variables in form of two-row matrices.
X'	Down-sampled random variable.
$\mathbf{V}, \mathbf{W}, \mathbf{Q}$	One-column vectors representing X, Y, and Z in the given order.
$\lvert\mathbf{V}\rvert$	Cardinality of vector $\mathbf{V}$.
$\mathbf{V} \odot \mathbf{W}$	Element-wise product between two vectors.
$\mathcal{F}\{\mathbf{V}\}$	Fourier Transform of vector $\mathbf{V}$.
$\mathcal{F}^{-1}\{\mathbf{V}\}$	Inverse Fourier Transform of vector $\mathbf{V}$.

In the above definition, Ω is a sample space, the set of all possible outcomes. $\mathfrak{F}$ is an event space, where an event is a set of outcomes in the sample space. $\mathbb{P}$ represents a probability function, that assigns each event in the event space a probability.

The image of Ω under X is denoted with $\operatorname{Im} X$, and it is the set of values taken by X, with positive probability.

Given a random variable X, we define the cumulative distribution function of X as $F_X(x) = \mathbb{P}(X \leq x)$, and its expected value (expectation) as $\mathbb{E}(X) = \sum_{x \in \operatorname{Im} X} x \cdot \mathbb{P}(X = x)$.

Throughout the paper, we will use a two-row matrix to represent the mapping between the obtainable values (sorted in increasing order), and their respective probabilities:

$$X \sim \begin{bmatrix} x_1 & x_2 & \cdots & x_n \\ \mathbb{P}(X = x_1) & \mathbb{P}(X = x_2) & \cdots & \mathbb{P}(X = x_n) \end{bmatrix}, \tag{1}$$

where the symbol $\sim$ denotes that the random variable X has the probability distribution described by the two-row matrix, and n is the cardinality of $\operatorname{Im} X$.

▶ **Definition 2** (Usual Stochastic order [26]). *Two random variables X and Y, with cumulative distribution functions F_X and F_Y, are said to be in the usual stochastic order, denoted as $X \succeq Y$, if and only if $\forall x$, $F_X(x) \leq F_Y(x)$.*

▶ **Definition 3** (Independence). *Two (discrete) random variables X and Y are independent if the pair of events $\{X = x\}$ and $\{Y = y\}$ are independent for all $x, y \in \mathbb{R}$. Formally,*

$$\mathbb{P}(X = x, Y = y) = \mathbb{P}(X = x)\mathbb{P}(Y = y) \qquad \text{for all} \quad x, y \in \mathbb{R}.$$

▶ **Definition 4** (Convolution or sum of random variables). *If X and Y are independent discrete random variables on $(\Omega, \mathfrak{F}, \mathbb{P})$, then $Z = X + Y$ has the mass function*

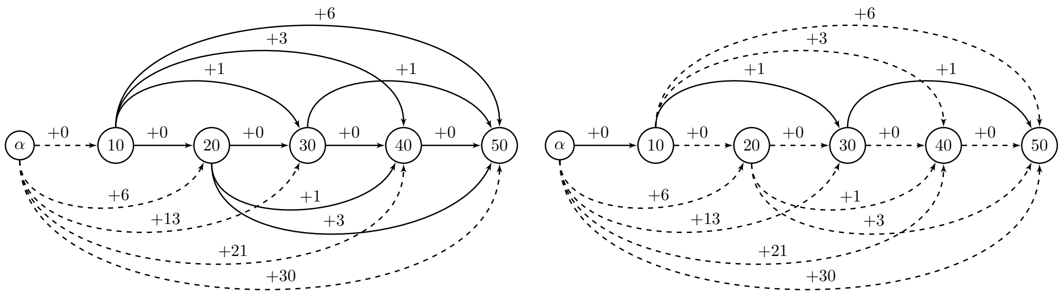
$$\mathbb{P}(Z = z) = \sum_{x=-\infty}^{\infty} \mathbb{P}(X = x)\mathbb{P}(Y = z - x) \qquad \text{for } z \in \mathbb{R}.$$

▶ **Definition 5** (Element-wise product). *The element-wise product (also known as, Hadamard product, point-wise product, and Schur product) between two matrices $\mathbf{A}$ and $\mathbf{B}$ of the same dimension $m \times n$ is denoted as $\mathbf{A} \odot \mathbf{B}$.*

▶ **Definition 6** (Discrete convolution between two vectors). *Given two vectors $\mathbf{V}$ and $\mathbf{W}$ of equal length n, the discrete convolution between the two vectors is defined as*

$$(\mathbf{V} * \mathbf{W}) = \sum_{i=1}^{n} \mathbf{V}(i)\mathbf{W}(n - i) = \sum_{i=1}^{n} \mathbf{W}(i)\mathbf{V}(n - i),$$

where $\mathbf{V}(i)$ (and $\mathbf{W}(i)$) is the i-th element of the vector $\mathbf{V}$ (of $\mathbf{W}$).



(a) Potential paths for down-sampling. **(b)** Optimal path (solid).

Figure 1 Graph representation of the additional expectations from the running example.

The majority of the above definitions are available in the book [13]. Definition 2 is cited from [26], while it is described by Diaz et al. [12] as the first-order stochastic dominance. In Table 1, we provide the list of the most important and frequently used symbols in the paper.

3 Down-sampling of Random Variables

We divide this section into two main parts. Section 3.1 describes the problem of optimal down-sampling with respect to minimising the probabilistic expectation. We solve this problem by proposing an algorithm that uses a dynamic programming approach. Section 3.2 proposes a linear down-sampling algorithm, which exhibits an improved time complexity compared to the optimal algorithm, at the cost of an increased over-approximation in some cases.

3.1 Optimal down-sampling of random variables

▶ **Problem 1** (Optimal down-sampling of random variables). *Given a discrete random variable X where n is the cardinality of* $\operatorname{Im} X$*, down-sample the random variable to cardinality s, $s < n$, such that the expectation of the derived variable X' is minimised, while $X' \succeq X$.*

Upon down-sampling, the probabilistic expectation of the down-sampled random variable X' is often larger than the expectation of the original random variable X since the probability mass in X' has to be shifted to the larger values in order to account that X' upper bounds X (Definition 2). For this reason, Maxim et al. [19] showed that expectation can be regarded as the metric of pessimism when the random variable is down-sampled from its original cardinality to a lower one. Many down-sampling algorithms were proposed in the state-of-the-art to reduce the probabilistic expectation upon down-sampling (e.g. Max-k re-sampling, Pessimism reduce re-sampling [19], etc.). In this section, we show how the random variable can be down-sampled to cardinality s such that its probabilistic expectation is less than or equal to the expectation of any other potential down-sampling of the same cardinality. To better clarify the problem, we introduce the following running example.

▶ Example 1. Initially, the random variable X contains the following values with the assigned probabilities (expressed with the notation of Equation (1)):

$$X \sim \begin{bmatrix} 10 & 20 & 30 & 40 & 50 \\ 0.6 & 0.1 & 0.1 & 0.1 & 0.1 \end{bmatrix}, \qquad \mathbb{E}(X) = 20. \tag{2}$$

The goal is to derive the down-sampled variable X' of cardinality 3 such that $\forall x \in \operatorname{Im} X$, $F_{X'}(x) \leq F_X(x)$ and such that $\mathbb{E}[X']$ is minimised.

Problem 1 can be solved in two steps.

Step 1. For any two values $k, l \in \operatorname{Im} X$ with $k < l$, compute the additional probabilistic expectation that would be present in the resulting variable X' if $k, l \in \operatorname{Im} X'$ and no other value between k and l is present in $\operatorname{Im} X'$.

Regarding the running example, the additional expectations can be represented with the weighted graph (see Figure 1a), such that vertices represent values, and solid edges represent additional expectation considering the case when two connected values are selected to be in the down-sampled variable.

Step 2. Use dynamic programming to select s values prior the last value of X, such that the total additional expectation is minimised. Considering the graph representation (see Figure 1a) the problem is equivalent to the problem of finding the path of size s from the artificial zero node (denoted α), to the last node of the graph. The zero node is added such that path can start from any node, i.e. any value in X, without necessarily starting from the first value in $\operatorname{Im} X$.

In the remainder of the section, we formalise different terms that are used in the algorithm, which is followed by the pseudo code and the algorithm description.

▶ **Definition 7.** *Additional expectation $\mathbb{E}_{k,l}(X)$ that is introduced in X' when only l is present in $\operatorname{Im} X'$ within interval $(k, l]$, is defined with the following equation*

$$\mathbb{E}_{k,l}(X) = l \sum_{i=k+1}^{l} \mathbb{P}(X = i) - \sum_{i=k+1}^{l} i\,\mathbb{P}(X = i) = \sum_{i=k+1}^{l} (l - i)\mathbb{P}(X = i). \tag{3}$$

In the first equality term of Equation (3), we consider that $\mathbb{E}_{k,l}(X)$ is equal to the difference between the following two terms:

1. $l \sum_{i=k+1}^{l} \mathbb{P}(X = i)$, which is the expectation in the interval $(k, l]$ when all the values between k and l are removed. Meaning that for the removed values their probabilities are accumulated to the value l in order to preserve safety (see Definition 2).
2. $\sum_{i=k+1}^{l} i\,\mathbb{P}(X = i)$, which is the original expectation of X within the interval $(k, l]$.

The expression is further simplified in order to account for only one traversal from k to l. From the example in Figure 1a, $\mathbb{E}_{20,40} = (40 - 30) \cdot 0.1 = +1$.

Considering the graph representation of the problem, where nodes represent values from X, and edges represent the potential selection, a weight on the edge between two nodes, k and l, is equal to the additional expectation $\mathbb{E}_{k,l}(X)$ defined in Equation (3). Thus, Problem 1 is analogous to the problem of finding a path of size s to the last value from X such that cost (expectation) is minimised. We use plus symbols in the weights to express the concept of *additional* expectation that is introduced in the down-sampled random variable (RV) compared to the original one. To solve such problem, we propose Algorithm 1.

3.1.1 Algorithm description

Algorithm 1 takes the input variables: (i) X, which is the discrete random variable to be down-sampled, and (ii) s, which is the cardinality for down-sampling. Additionally to the down-sampled variable X', the algorithm also outputs the difference E_n between the expectation of the original random variable and the down-sampled one.

Algorithm 1 Optimal down-sampling of a discrete random variable.

```
   Data: Discrete random variable $X$, cardinality $s$ for down-sampling
   Result: Down-sampled random variable $X'$
1  function optimalDS($X, s$):
2      $n \leftarrow \max(\mathrm{Im}\, X)$
3      $\alpha = \min(\mathrm{Im}\, X) - 1$
4      for $l \in \mathrm{Im}\, X$ do
5          $E_l \leftarrow \mathbb{E}_{\alpha,l}(X)$
6          $V_l \leftarrow \emptyset$
7      end
8      for $i \leftarrow 1$ to $s$ by $1$ do
9          for $l \in \mathrm{Im}\, X$ do
10             $E'_l \leftarrow +\infty$
11             $V'_l \leftarrow \emptyset$
12         end
13         for $k, l \in \mathrm{Im}\, X : k < l$ do
14             if $E'_l > E_k + \mathbb{E}_{k,l}$ then
15                 $E'_l \leftarrow E_k + \mathbb{E}_{k,l}$
16                 $V'_l \leftarrow V_k \cup k$
17             end
18         end
19         for $l \in \mathrm{Im}\, X$ do
20             $E_l \leftarrow E'_l$
21             $V_l \leftarrow V'_l$
22         end
23     end
24     $X' \leftarrow$ RV such that $\mathrm{Im}\, X' = V_n$ and $\mathbb{P}(X' = x') = \sum_{x=\mathrm{prec}(x')+\epsilon}^{x'} \mathbb{P}(X = x)$, where
         $\mathrm{prec}(x')$ is the value preceding $x'$ from $\mathrm{Im}\, X'$, and $\mathrm{prec}(\min(\mathrm{Im}\, X')) = \min(\mathrm{Im}\, X)$
25 return $X', E_n$
```

In line 3, the algorithm computes the artificial value α which represents the first value that precedes the minimum value from $\mathrm{Im}\, X$ (see Figure 1a). Then, in line 5, α is used to compute the minimum expectation E_l that is imposed if we select some value l from X as the first selected for the down-sampled variable, without anymore considering the values that precede l. The additional expectation of performing such a step is equal to $\mathbb{E}_{\alpha,l}(X)$ (see Figure 1a). Since at the beginning, no value is still selected, for each value l from $\mathrm{Im}\, X$, we initialise V_l which is the set of values whose selection yields the minimum expectation until the value l. In lines 8–23, the algorithm computes and updates the minimum expectations that can be imposed until any value l from X, upon selection of at most i values that precede l. Upon accounting for each new selection, the minimum expectation until each value l is updated (line 20), together with the set V_l of selected values that yield the minimised expectation (line 21). After doing this step s times, the V_n represents the set that yields E_n, while E_n represents the minimum additional expectation until the last value n (line 2) upon selection of s values prior the last one. This is equivalent to the dynamic programming problem: $E[i+1, l] = \min_{k<l}\{E[i,k] + \mathbb{E}_{k,l}\}$, where $E[a,x]$ is the minimum additional expectation until value x after a selections.

3.1.2 Algorithm correctness

We prove that the algorithm minimises the additional expectation by using induction.

Proof. By induction.
Induction hypothesis: For $l \in \operatorname{Im} X$, after the i-th iteration of the *for* loop at line 8, E_l represents the minimum additional expectation that can be produced until value l, considering at most i selected values prior l, while V_l represents the set of selected i values whose joint expectation yields E_l.
Base case: Before the loop at line 8 starts, the number of iterations is $i = 0$, E_l is equal to the expectation $\mathbb{E}_{\alpha,l}(X)$ which is also the minimum additional expectation imposed without selecting any value that precedes l, as follows from Equation (3). Also, V_l is an empty set, representing that no value preceding l is selected. Thus, the base case holds.
Inductive step: We now prove that the induction hypothesis holds at the end of $i+1$ iteration of the for loop (lines 8–23). At the beginning of the $i+1$ iteration, for every value $l \in \operatorname{Im} X$, Algorithm 1 identifies the respective preceding value k such that the expectation of selecting k prior to l yields the minimum expectation (lines 13–18). This is the case because the expectation is minimised over the all possible choices of $k : k < l \wedge k \in \operatorname{Im} X$ (line 13). Also, it follows from the induction hypothesis that until value k the expectation is minimised with at most i selected values before k. Then, the derived expectation E'_l at the end of the loop (line 18) is the minimum expectation that can be produced with at most $i+1$ selected values until l, where k is the additional $(+1)$ selected value. This further means that E_l also represents the demanded minimum possible expectation since after line 20, $E_l = E'_l$. Analogically, the same holds for the set of selected values V_l that yields E_l since the set is computed as the union of 1) values which yield the minimum expectation E_k, as follows from the induction hypothesis, and 2) value k which is the additional selected value that yields the minimum E_l as shown previously. Finally, after s iterations, it follows that the algorithm computes the minimum additional expectation E_n which is the result of selecting s values prior the last one, n. Analogously, it holds that V_n stores the selected values whose selection yields the minimum added expectation. Thus, the random variable X' (line 24) resulting from the selected values in V_n, yields the minimum additional expectation E_n among the other possible random variables of the same cardinality, constructed from values in X, also upper-bounding X. This concludes the proof. ◀

3.1.3 Time complexity

The worst-case time complexity of Algorithm 1 is $\mathcal{O}(n^3)$ since there are at most $((n+1)n)/2$ expectations of form $\mathbb{E}_{k,l}$ to be computed, and for each such computation, Equation 3 has linear complexity, which finally results in $\mathcal{O}((n+1)n)\mathcal{O}(n) = \mathcal{O}(n^3)$.

3.2 Linear Down-sampling

In this subsection, we propose Algorithm 2 for down-sampling of random variables, that exhibits linear time complexity. The main idea behind the algorithm is uniform down-sampling of a probability distribution. The algorithm starts from the first value of the original random variable and it iterates until the last one (line 6). It assigns the subset of values from $\operatorname{Im} X$ such that the cumulative probability between consecutive selected values in X' is as uniform as possible, under the condition of $X' \succeq X$. The considered terms are:
- P^{un} – unassigned probability sum, i.e., part of the distribution that is not present in the current version of the down-sampled variable X',
- s – number of values to be assigned to the down-sampled variable X',
- p_δ – probability sum threshold, which controls that the number of values in the resulting down-sampled variable does not exceed the predefined cardinality s,

Algorithm 2 Linear down-sampling of a discrete random variable.

```
   Data: Discrete random variable $X$, cardinality $s$ for down-sampling
   Result: Down-sampled random variable $X'$
 1 function linearDS($X, s$):
 2     $P^{un} \leftarrow 1$ // sum of the unassigned probability values from $X$ to $X'$
 3     $p_\delta \leftarrow P^{un}/s$
 4     $\mathrm{Im}\, X' \leftarrow \emptyset$
 5     $p_\Sigma \leftarrow 0$
 6     for $l \in \mathrm{Im}\, X$ in an increasing order do
 7         $p_\Sigma \leftarrow p_\Sigma + \mathbb{P}(X = l)$
 8         $P^{un} \leftarrow P^{un} - \mathbb{P}(X = l)$
 9         if $p_\Sigma \geq p_\delta$ then
10             $\mathrm{Im}\, X' \leftarrow \mathrm{Im}\, X' \cup l$ such that $\mathbb{P}(X = l) = p_\Sigma$
11             $s \leftarrow s - 1$
12             $p_\delta \leftarrow P^{un}/s$
13             $p_\Sigma \leftarrow 0$
14         end
15     end
16     $X' \leftarrow$ RV such that $\mathrm{Im}\, X' = V_n$ and $\mathbb{P}(X' = x') = \sum_{x=\mathrm{prec}(x')+\epsilon}^{x'} \mathbb{P}(X = x)$, where
         $\mathrm{prec}(x')$ is the value preceding $x'$ from $\mathrm{Im}\, X'$, and $\mathrm{prec}(\min(\mathrm{Im}\, X')) = \min(\mathrm{Im}\, X)$
17 return $X'$
```

- p_Σ – probability sum from the last selected value until the currently observed one,
- $\mathrm{Im}\, X'$ – set of selected values for X'.

Given the above terms, in each iteration the algorithm assigns the currently observed value $l \in \mathrm{Im}\, X$ to the down-sampled variable X', but only in case the probability sum p_Σ, from the last selected value in X', exceeds the probability threshold p_δ (line 9). Probability threshold p_δ maintains uniformity of the probability distribution and does not allow that more than s values are selected within X'. This is achieved by constant re-computation of the still unassigned probability P^{un}, whenever a new value is assigned to X' (line 10). Also, whenever a new value is assigned to X', value p_Σ is set to zero to account for the fact that no new value and respective probability is assigned from the last assignment (line 13). In case when l is not selected, p_Σ is updated with the probability resulting from l itself (line 8). At the end, the algorithm returns the down-sampled random variable X' (line 17).

3.3 Description of the algorithms using the running example

Given the random variable X from Example 1, and the process of down-sampling to the cardinality of 3, Algorithm 1 selects the values 10, 30, and 50 as depicted in Figure 1b with solid arrows above the values. This is the shortest path of size 2 until the last value 50, with non-zero probability. Note that in order to select three values, Algorithm 1 should be invoked with $s = 2$ since the largest value from $\mathrm{Im}\, X$ will always be selected (line 24).

Algorithm 2 selects the same two values using a different approach. For the desired cardinality $s = 3$, it first computes that the probability sum threshold p_δ is equal to $1/3 = 0.333$ (line 3). Then, by trying to uniformly distribute the cumulative probability sum, it immediately at value 10 (lines 6 – 8) identifies that the threshold is exceeded since $0.3 < \mathbb{P}(X = 10)$ (line 9), and adds 10 to the down-sampled variable (line 10). By recomputing the threshold (line 12), for the remainder of the distribution, it derives the new threshold $p_\delta = (1 - 0.6)/2 = 0.2$ since 0.4 is the unassigned probability sum, and there are two more

$$\text{(a)}\quad \begin{tikzcd}(X,Y) \arrow[r, Rightarrow, "\text{addition}", "\text{multiplication}"'] & Z'' \arrow[r, Rightarrow, "\text{sorting}", "\text{sorting}"'] & Z' \arrow[r, Rightarrow, "\text{normalisation}", "\text{addition}"'] & Z\end{tikzcd}$$

$$\text{(b)}\quad \begin{tikzcd}(X,Y) \sim (\mathbf{V},\mathbf{W}) \arrow[r, "\mathcal{F}\{\cdot\}"] & (\hat{\mathbf{V}},\hat{\mathbf{W}}) \arrow[r, "\text{element-wise product}"] & \hat{\mathbf{Q}} \arrow[r, "\mathcal{F}^{-1}\{\cdot\}"] & \mathbf{Q} \sim Z\end{tikzcd}$$

Figure 2 Overview of (a) linear, and (b) circular convolutions for computing $X + Y = Z$.

values to be selected. At $l = 20$, its probability is $\mathbb{P}(X = 10) = 0.1$, thus the threshold is not reached, but in the next iteration $l = 30$, it is (line 9, i.e. $0.1+0.1 \geq 0.2$). At the end, the sum of the remaining unassigned probability mass is equal to $P^{un} = (1-0.6-0.2) = 0.2$, and there is only one value to be selected. This means that the algorithm selects the last value 50 and assigns to it the probability of 0.2. For both algorithms, $\mathbb{E}(X') = 22$ and $X' = \left[\begin{smallmatrix}10 & 30 & 50\\ 0.6 & 0.2 & 0.2\end{smallmatrix}\right]$.

4 Efficient convolution

Probabilistic timing analysis techniques suffer from a large time complexity that is essentially attributable to the linear convolution operator [10].

▶ **Problem 2** (Efficient convolution of random variables). *Improve the efficiency of computing the exact result of the convolution between two random variables.*

In the following, we describe different ways of computing the sum of two random variables, and we present different improvements that can be used to reduce their time complexity.

Let X and Y be independent discrete random variables, such that

$$X \sim \begin{bmatrix} x_1 & \cdots & x_n \\ \mathbb{P}(X = x_1) & \cdots & \mathbb{P}(X = x_n) \end{bmatrix}, \quad Y \sim \begin{bmatrix} y_1 & \cdots & y_m \\ \mathbb{P}(Y = y_1) & \cdots & \mathbb{P}(Y = y_m) \end{bmatrix}.$$

The addition $X + Y$ of the two random variables is the random variable Z whose probability distribution is characterised by the convolution between the probability distributions of X and Y, and it is defined as:

$$\mathbb{P}(Z = z) = \sum_{k=0}^{+\infty} \mathbb{P}(X = k)\mathbb{P}(Y = z - k), \quad \operatorname{Im} Z = \{z | \forall x \in \operatorname{Im} X, \forall y \in \operatorname{Im} Y, z = x + y\}.$$

Linear convolution. The linear convolution (also known as *canonical convolution*) is performed in three steps, as described by Milutinović et al. [22]. We show this in Figure 2(a), where the symbol $\overset{v}{\underset{\mathrm{p}}{\Longrightarrow}}$ denotes that operation v is performed on the values of the random variables (indicated above the arrow), and that operation p is applied to the respective probability distributions (indicated below the arrow). Reading the figure from the left, we start with X and Y, with their respective probability distributions. Then, the algebraic addition of each possible pair of values in $\operatorname{Im} X$ and in $\operatorname{Im} Y$ is computed, and for each computed value the multiplication of the respective probability distribution is computed, obtaining the variable Z''. Then, the values in $\operatorname{Im} Z''$ are sorted in increasing order, and the associated probabilities are sorted accordingly, thus deriving the variable Z'. At the end, a normalisation is performed on the values of Z' such that the repeated values are combined, and their respective probabilities are summed. Such algorithm has a time complexity of $\mathcal{O}(n \cdot m)$, where n and m are the cardinalities of $\operatorname{Im} X$ and $\operatorname{Im} Y$, respectively. For more details, refer to paper by Milutinović et al. [22].

Circular convolution. To solve Problem 2, we build upon the idea of *circular convolution*. There are many mathematical sources that explain the benefit of circular over the linear convolution [15, 23]. The circular convolution idea is shown in Figure 2(b). Starting from X and Y, we first compute vectors $\mathbf{V}$ and $\mathbf{W}$, respectively, such that vector indexes represent values, while the vector elements represent probabilities of the corresponding random variable. Note that in Figure 2(b), we use $\xrightarrow{e}$ to represent that operation e is performed on the vector elements. On the vectors $\mathbf{V}$ and $\mathbf{W}$, we perform the Fourier Transformation operation, obtaining the new vectors $\hat{\mathbf{V}}$ and $\hat{\mathbf{W}}$. Then, we perform the element-wise product between the transforms, deriving the transform $\hat{\mathbf{Q}}$. Next, we compute the inverse Fourier transformation of $\hat{\mathbf{Q}}$, deriving the vector $\mathbf{Q}$, which characterises random variable Z, i.e., indexes of $\mathbf{Q}$ represent values of Z, while vector elements represent probabilities of Z. In Figure 2, this is denoted with $\mathbf{Q} \sim Z$.

The above-described process of computing the Fourier and inverse Fourier transformations, together with the element-wise product of two transforms, is known as *circular convolution*, and it has a time complexity of $\mathcal{O}(d \log(d))$, where $d = \max(\operatorname{Im} Z) - \min(\operatorname{Im} Z)$. We discuss the whole process formally, in more detail, in the remaining part of this section.

4.1 Formal description of the circular convolution of random variables

Let us start from the convolution theorem [1,3,5]. In the following formulation of the theorem we narrow it to vectors although it holds for more complex mathematical entities.

▶ **Theorem 1** (Fourier's convolution theorem [24]). *The Fourier transform of the convolution of two vectors* $\mathbf{V}$ *and* $\mathbf{W}$ *is equal to the element-wise product of the Fourier transforms of the two vectors, i.e.*

$$\mathcal{F}\{(\mathbf{V} * \mathbf{W})\} = \mathcal{F}\{\mathbf{V}\} \odot \mathcal{F}\{\mathbf{W}\}. \tag{4}$$

In the above equation, $\odot$ represents linearly complex element-wise multiplication between vectors $\mathcal{F}\{\mathbf{V}\}$ and $\mathcal{F}\{\mathbf{W}\}$ of Fourier coefficients. To compute the convolution of $\mathbf{V}$ and $\mathbf{W}$, we can compute the inverse Fourier transform of the right side of the equality, and this process is known as the circular convolution. However, the result of the circular convolution $\operatorname{cconv}(\mathbf{V}, \mathbf{W})$ of two vectors $\mathbf{V}$ and $\mathbf{W}$ is equal to the result of the corresponding linear convolution, (as presented in [15, 21]) when the following equations hold

$$\begin{aligned} \operatorname{cconv}(\mathbf{V}, \mathbf{W}) = \mathcal{F}^{-1}\left\{\hat{\mathbf{V}} \odot \hat{\mathbf{W}}\right\}, \text{ where } \hat{\mathbf{V}} &= \mathcal{F}\{\mathbf{V}^\frown 0_v\}, \\ \hat{\mathbf{W}} &= \mathcal{F}\{\mathbf{W}^\frown 0_w\}, \\ v &= \operatorname{nptwo}(|\mathbf{V}| + |\mathbf{W}| - 1) - |\mathbf{V}|, \\ w &= \operatorname{nptwo}(|\mathbf{V}| + |\mathbf{W}| - 1) - |\mathbf{W}|, \end{aligned} \tag{5}$$

where $\operatorname{nptwo}(a)$ is a function that returns the first power of two greater than or equal to some value $a \in \mathbb{N}$, and $\frown$ is a concatenation operator. In the above equation, vectors $\mathbf{V}$ and $\mathbf{W}$ first need to be zero-padded to the same size, and that size must be greater than or equal to the the sum of their respective sizes minus one, as shown by Langton and Levin [15]. This size, in the equation, is represented by the following term $|\mathbf{V}| + |\mathbf{W}| - 1$. The zero-padding of the vectors is performed with the vector concatenation operator ($\frown$). This operator is used between the desired vector (e.g. $\mathbf{V}$) and the zero-column vector (e.g. 0_v). The concatenation of the zero-column vector to the desired vector leads to the desired zero-padding. Similar computations are defined for the vector $\mathbf{W}$ as well, considering the zero-column vector 0_w.

Additionally, in the equation, the zero-padding is increased to reach the size that is equal to the first power of two that succeeds $|\mathbf{V}| + |\mathbf{W}| - 1$. This step, of computing the $\mathrm{nptwo}(\cdot)$ function, is performed in order to allow for the linearithmic time complexity that can be achieved by using the Discrete Fast Fourier Transformation, known also as Cooley-Tukey algorithm [9].

Without the loss of generality, for the sake of the simplified equations and the running example, suppose that $\{0, 1, 2, \dots, b\}$, where $b \in \mathbb{N}$, is the support of discrete random variables X and Y.

Considering sum Z of random variables X and Y, it can be computed using Equation (5) such that for discrete random variables X and Y, their probabilities are represented as the elements of vectors $\mathbf{V}$ and $\mathbf{W}$ respectively, as shown in the following equations.

$$\begin{aligned}
&\mathbf{V} = (v_j) = \mathbb{P}(X = j) \wedge j \in \{0, \dots, m_X\} \text{ , where } m_X = \max(\{x \mid x \in \operatorname{Im} X\}),\\
&\mathbf{W} = (w_j) = \mathbb{P}(Y = j) \wedge j \in \{0, \dots, m_Y\} \text{ , where } m_Y = \max(\{y \mid y \in \operatorname{Im} Y\}),\\
&\mathbf{Q} = \mathrm{cconv}(\mathbf{V}, \mathbf{W}) = (q_j) \text{ and } Z \sim \begin{bmatrix} 0 & \dots & j & \dots & m_X + m_Y \\ q_0 & \dots & q_j & \dots & q_{m_X+m_Y} \end{bmatrix}.
\end{aligned} \tag{6}$$

▶ Example 2.

$$X \sim \begin{bmatrix} 200 & 300 \\ 0.6 & 0.4 \end{bmatrix} \qquad Y \sim \begin{bmatrix} 150 & 200 \\ 0.6 & 0.4 \end{bmatrix} \qquad Z \sim \begin{bmatrix} 350 & 400 & 450 & 500 \\ 0.36 & 0.24 & 0.24 & 0.16 \end{bmatrix}$$

$$\mathbf{V} = \begin{array}{c} \begin{bmatrix} 0 \\ \vdots \\ 0.6 \\ 0 \\ \vdots \\ 0.4 \end{bmatrix} \end{array} \begin{array}{l} 0 \\ \\ 200 \\ 201 \\ \\ 300 \end{array} \qquad \mathbf{W} = \begin{array}{c} \begin{bmatrix} 0 \\ \vdots \\ 0.6 \\ 0 \\ \vdots \\ 0.4 \end{bmatrix} \end{array} \begin{array}{l} 0 \\ \\ 150 \\ 151 \\ \\ 200 \end{array} \qquad \mathbf{Q} = \begin{array}{c} \begin{bmatrix} \vdots \\ 0.36 \\ \vdots \\ 0.24 \\ \vdots \\ 0.24 \\ \vdots \\ 0.16 \\ \vdots \end{bmatrix} \end{array} \begin{array}{l} \\ 350 \\ \\ 400 \\ \\ 450 \\ \\ 500 \\ 511 \end{array}$$

Starting from the random variable X, in Equation (6), we first define vector V such that its index j represents all the values in the domain of X, starting from 0 until the last value $m_X \in X$ that has non-zero probability of occurrence. Then, the j-th value v_j in V represents the probability $P(X = j)$. Similar is performed for vector W but considering the random variable Y. Then, using Equation (5) we compute the circular convolution of V and W thus deriving vector Q, whose j-th value is equal to the probability $P(X + Y = j)$. Finally, as follows from Theorem 1 and Equation (6), the vector Q contains the combined information on probability values from $\operatorname{Im} X + Y$.

In the above example, the omitted numbers represent probabilities equal to zero. The majority of computations between those zeros can be avoided, while still maintaining the exact computation of the final result. This is the problem that we solve in the following subsection by proposing two methods for vector reductions such that the computations derive the exact result.

4.2 Fast and efficient computation of the exact result

Compared to the solution from Equation (6), it is possible to derive the resulting random variable Z with more efficient computations.

▶ **Improvement 1.** *Reducing the vector sizes using the greatest common divisor.*

In this solution, the improvement is made by using the greatest common divisor among the values from $\operatorname{Im} X$ and $\operatorname{Im} Y$. As follows from Definition 4, a value with zero probability of occurrence cannot lead to the non-zero probability value in Z. The same holds for the analogous Equation (6). Thus, we can improve the computation by considering the minimum quantum that considers all non-zero values from both equations, as follows:

$$\delta = \mathrm{GCD}(\operatorname{Im} X \cup \operatorname{Im} Y), \tag{7}$$

$$\mathbf{V} = (v_j) = \mathbb{P}(X = j) \wedge j \in \{0, \dots, m_X\} \text{ , where } m_X = 1/\delta \cdot \max(\{x \mid x \in \operatorname{Im} X\}), \tag{8}$$

$$\mathbf{W} = (w_j) = \mathbb{P}(Y = j) \wedge j \in \{0, \dots, m_Y\} \text{ , where } m_Y = 1/\delta \cdot \max(\{y \mid y \in \operatorname{Im} Y\}), \tag{9}$$

$$\mathbf{Q} = \mathrm{cconv}(\mathbf{V}, \mathbf{W}) = (q_j), \tag{10}$$

$$Z \sim \begin{bmatrix} 0 \cdot \delta & \dots & j \cdot \delta & \dots & (m_X + m_Y) \cdot \delta \\ q_0 & \dots & q_j & \dots & q_{m_X + m_Y} \end{bmatrix}. \tag{11}$$

With the above equation, the sum Z of X and Y from Example 2 would look as follows:

$$\delta = 50, \ \mathbf{V} = \begin{array}{cl} \begin{bmatrix} 0 \\ \vdots \\ 0.6 \\ 0 \\ 0.4 \end{bmatrix} & \begin{array}{l} 0 \\ \\ 4 \\ 5 \\ 6 \end{array} \end{array}, \ \mathbf{W} = \begin{array}{cl} \begin{bmatrix} 0 \\ \vdots \\ 0.6 \\ 0.4 \end{bmatrix} & \begin{array}{l} 0 \\ \\ 3 \\ 4 \end{array} \end{array}, \ \mathbf{Q} = \mathrm{cconv}(\mathbf{V}, \mathbf{W}) = \begin{array}{cl} \begin{bmatrix} \vdots \\ 0.36 \\ 0.24 \\ 0.24 \\ 0.16 \\ \vdots \end{bmatrix} & \begin{array}{l} \\ 7 \\ 8 \\ 9 \\ 10 \\ 15 \end{array} \end{array} \tag{12}$$

$$Z \sim \begin{bmatrix} 350 = 7 \cdot 50 & 400 = 8 \cdot 50 & 450 = 9 \cdot 50 & 500 = (6 + 4) \cdot 50 \\ 0.36 & 0.24 & 0.24 & 0.16 \end{bmatrix}.$$

The essence of the above described transformation is to change the base metric unit such that unnecessary computations are avoided. Compared to Example 2, the vector sizes are roughly 30 times less in the above equation, which also propagates to computation time.

▶ **Improvement 2.** *Reducing the vector sizes by removing the starting zero intervals.*

In this improvement we focus on the starting values of the vectors $\mathbf{V}$ and $\mathbf{W}$. Consider Example 2 and the follow-up Equation (12), both vectors $\mathbf{V}$ and $\mathbf{W}$ start with probabilities equal to zero, followed by more zero probabilities that represent values that are not in $\operatorname{Im} X$ and $\operatorname{Im} Y$. We show that both vectors can be reduced by ignoring starting zero intervals, thus starting from the probabilities of the minimum values in $\operatorname{Im} X$ and $\operatorname{Im} Y$, without losing computation precision. Let us start from random variables X and Y:

$$X \sim \begin{bmatrix} x_1 & \dots & x_n \\ \mathbb{P}(X = x_1) & \dots & \mathbb{P}(X = x_n) \end{bmatrix}, \quad Y \sim \begin{bmatrix} y_1 & \dots & y_m \\ \mathbb{P}(Y = y_1) & \dots & \mathbb{P}(Y = y_m) \end{bmatrix}. \tag{13}$$

▶ **Proposition 2.** *Given two random variables X and Y, the convolution $X + Y$ is*

$$X{+}Y \sim \begin{bmatrix} x_1 + y_1 \\ 1 \end{bmatrix} + \begin{bmatrix} x_1 - x_1 & \dots & x_n - x_1 \\ \mathbb{P}(X = x_1) & \dots & \mathbb{P}(X = x_n) \end{bmatrix} + \begin{bmatrix} y_1 - y_1 & \dots & y_m - y_1 \\ \mathbb{P}(Y = y_1) & \dots & \mathbb{P}(Y = y_m) \end{bmatrix}. \tag{14}$$

Proof.

$$\begin{aligned} &\begin{bmatrix} x_1 + y_1 \\ 1 \end{bmatrix} + \begin{bmatrix} x_1 - x_1 & \dots & x_n - x_1 \\ \mathbb{P}(X = x_1) & \dots & \mathbb{P}(X = x_n) \end{bmatrix} + \begin{bmatrix} y_1 - y_1 & \dots & y_m - y_1 \\ \mathbb{P}(Y = y_1) & \dots & \mathbb{P}(Y = y_m) \end{bmatrix} \\ &= \begin{bmatrix} x_1 \\ 1 \end{bmatrix} + \begin{bmatrix} x_1 - x_1 & \dots & x_n - x_1 \\ \mathbb{P}(X = x_1) & \dots & \mathbb{P}(X = x_n) \end{bmatrix} + \begin{bmatrix} y_1 \\ 1 \end{bmatrix} + \begin{bmatrix} y_1 - y_1 & \dots & y_m - y_1 \\ \mathbb{P}(Y = y_1) & \dots & \mathbb{P}(Y = y_m) \end{bmatrix} \\ &= \begin{bmatrix} x_1 & \dots & x_n \\ \mathbb{P}(X = x_1) & \dots & \mathbb{P}(X = x_n) \end{bmatrix} + \begin{bmatrix} y_1 & \dots & y_m \\ \mathbb{P}(Y = y_1) & \dots & \mathbb{P}(Y = y_m) \end{bmatrix} \sim X + Y \end{aligned}$$ ◀

The benefit of using Proposition 2 is observable when we want to generate vectors $\mathbf{V}$ and $\mathbf{W}$. Consider creating vectors from random variable $X \sim \left[\begin{smallmatrix} 1000 & 1001 \\ 0.4 & 0.6 \end{smallmatrix}\right]$ and $Y \sim \left[\begin{smallmatrix} 1005 & 1006 \\ 0.4 & 0.6 \end{smallmatrix}\right]$. Instead of generating vectors of sizes measured in thousand units, we can simply apply the proposition and derive the following term:

$$\left[\begin{smallmatrix} 1000 & 1001 \\ 0.4 & 0.6 \end{smallmatrix}\right] + \left[\begin{smallmatrix} 1005 & 1006 \\ 0.4 & 0.6 \end{smallmatrix}\right] = 1000 + 1005 + \left[\begin{smallmatrix} 0 & 1 \\ 0.4 & 0.6 \end{smallmatrix}\right] + \left[\begin{smallmatrix} 0 & 1 \\ 0.4 & 0.6 \end{smallmatrix}\right], \tag{15}$$

and therefore we can represent random variables with two vectors, each having size of two.

Based on Proposition 2 and Improvement 1, we introduce the following computations.

$$X' = X - x_1,\ Y' = Y - y_1,\ \delta = \mathrm{GCD}(\mathrm{Im}\, X' \cup \mathrm{Im}\, Y'), \tag{16}$$

$$\mathbf{V} = (v_j) = \mathbb{P}(X' = j) \wedge j \in [0, \ldots, m_X\} \text{ , where } m_X = 1/\delta \cdot \max(\{x \mid x \in \mathrm{Im}\, X'\}), \tag{17}$$

$$\mathbf{W} = (w_j) = \mathbb{P}(Y' = j) \wedge j \in [0, \ldots, m_Y\} \text{ , where } m_Y = 1/\delta \cdot \max(\{y \mid y \in \mathrm{Im}\, Y'\}), \tag{18}$$

$$\mathbf{Q} = \mathrm{cconv}(\mathbf{V}, \mathbf{W}) = (q_j), \tag{19}$$

$$Z \sim \begin{bmatrix} x_1 + y_1 + 0 \cdot \delta & \ldots & x_1 + y_1 + j \cdot \delta & \ldots & x_1 + y_1 + (m_X + m_Y) \cdot \delta \\ q_0 & \cdots & q_j & \cdots & q_{m_X + m_Y} \end{bmatrix}. \tag{20}$$

The above set of computations is almost the same as the one for the first improvement, but the major difference is that we use Proposition 2 and therefore compute the circular convolution only for variables X' and Y' which results in the furthermore reduced vectors $\mathbf{V}$ and $\mathbf{W}$. In Equation (20), the result of $X + Y$ is restored by applying the proposition $(x_1 + y_1 + j \times \delta)$. We show the vector size reduction in the running example:

$$x_1 + y_1 = 200 + 150 = 350,\ X' \sim \begin{bmatrix} 0 & 100 \\ 0.6 & 0.4 \end{bmatrix},\ Y' \sim \begin{bmatrix} 0 & 50 \\ 0.6 & 0.4 \end{bmatrix},\ \delta = 50,$$

$$\mathbf{V} = \begin{array}{c|c} \begin{bmatrix} 0.6 \\ 0 \\ 0.4 \end{bmatrix} & \begin{smallmatrix} 0 \\ 1 \\ 2 \end{smallmatrix} \end{array},\ \mathbf{W} = \begin{array}{cc} \begin{bmatrix} 0.6 \\ 0.4 \end{bmatrix} & \begin{smallmatrix} 0 \\ 1 \end{smallmatrix} \end{array},\ \mathbf{Q} = \mathrm{cconv}(\mathbf{V}, \mathbf{W}) = \begin{array}{cc} \begin{bmatrix} 0.36 \\ 0.24 \\ 0.24 \\ 0.16 \end{bmatrix} & \begin{smallmatrix} 0 \\ 1 \\ 2 \\ 3 \end{smallmatrix} \end{array},$$

$$Z \sim \begin{bmatrix} 350 = 350 + 0 \cdot 50 & 400 = 350 + 1 \cdot 50 & 450 = 350 + 2 \cdot 50 & 500 = 350 + 3 \cdot 50 \\ 0.36 & 0.24 & 0.24 & 0.16 \end{bmatrix}.$$

In this simple example we reduced the number of elements in vectors from 512 values (as in Example 2) to 4 values at most, still deriving the exact random variable Z.

Impact of the improvements on the probabilistic timing analysis. Improvements 1 and 2 may simplify and improve the efficiency of many analysis types that are characterised with the iterative computation of the increasing probabilistic distribution.

▶ Example 3 (Computation of the probability distribution). Consider the computation of the probabilistic distribution from two tasks (τ_1 and τ_2) whose probabilistic execution times are defined with the following random variables $C_1 \sim \left[\begin{smallmatrix} 1000 & 1001 \\ 0.4 & 0.6 \end{smallmatrix}\right]$ and $C_2 \sim \left[\begin{smallmatrix} 1005 & 1006 \\ 0.4 & 0.6 \end{smallmatrix}\right]$. If we want to compute the probabilistic distribution that involves 100 jobs of C_1 and 200 jobs of C_2, we can simply use Proposition 2 and derive the following result: $100 \cdot C_1 + 200 \cdot C_2 = 100 \cdot 1000 + 200 \cdot 1005 + S$, where random variable S is characterised with $\mathcal{F}^{-1}\left\{\bigodot_1^{100} \hat{\mathbf{V}}_1 \odot \bigodot_1^{200} \hat{\mathbf{V}}_2\right\}$.

Vectors $\hat{\mathbf{V}}_1$ and $\hat{\mathbf{V}}_2$ are the Fourier transforms of vectors that represent C_1 and C_2 (each having size of two). The important benefit is that the random variable S, which follows the summation of the deterministic values, can be derived from the inverse Fourier transformation of the vector of 600 elements $(2 \cdot 100 + 2 \cdot 200)$, compared to the naive method from Equation (6), where the final vector size would be 301600 elements.

This is a huge improvement in the computation efficiency, while the magnitude of space and time reduction becomes even greater with each new job that may be considered in the analysis.

▶ Example 4 (Computation of the deadline miss probability). Let us consider the very same two tasks (τ_1 and τ_2) from the previous example, and assume that we want to compute the deadline miss probability at time instant $D_2 = 2000$. Furthermore, τ_1 can be released at most two times until D_2, while τ_2 can be released at most once. Instead of performing the entire computation, we can first apply Proposition 2 and check where the image of the resulting distribution starts, i.e. what is the first value s with non-zero probability of occurrence in the final distribution. Thus we derive that $s = 2 \cdot 1000 + 1 \cdot 1005 = 3005$. Since the image of the summed distribution starts from 3005, and since $s > D_2$, this implies that $\mathbb{P}(C_1 + C_1 + C_2 > 2000) = 1$ and there is no need to perform convolutions. Such an improvement reduces the number of probabilistic computations for such types of problems.

4.3 Efficient repetitive convolutions

In timing analysis, there are often cases when one variable is summed multiple times, e.g. request bound function (RBF) of form $RBF_i(\Delta) = C_i \cdot \alpha(\Delta)$ where C_i is the worst-case execution time of a task, Δ is the time interval under consideration, and $\alpha(\Delta)$ represents a function that upper-bounds the number of arrivals of jobs of τ_i within some time interval of length Δ. It has been shown by Bozhko and Brandenburg [4] that RBF can be used to concretize many existing types of response-time analysis, under various taskset model assumptions, and thus it represents the fundamental computation in the analysis of real-time systems.

Analogously, as shown in many probabilistic timing analysis papers, e.g., [12, 19], probabilistic request bound can be computed by consecutive addition of the random variables that represent upper-bounded probabilistic execution time of a task. Similar computation is necessary in many other areas of real-time system analysis, e.g., probability of cache miss and hit, etc. Therefore, in this section we describe the algorithm that efficiently computes consecutive additions of the same random variable, based on the consecutive Hadamard product of the Fourier coefficients, using the powers of two[1].

Algorithm 3 computes the result of n convolutions of random variable X. This computation can be achieved by creating the sum vector $\hat{\mathbf{V}}_{sum} = \mathbf{1}$, of all values equal to one, (line 5) which is iteratively multiplied n times with the Fourier Transform $\mathcal{F}\{\mathbf{V}\}$ of vector $\mathbf{V}$, where $\mathbf{V}$ characterises X (line 4). However, instead of that, we can perform fewer multiplications by constantly computing the power vector $\hat{\mathbf{V}}'$, which is initially set to $\mathcal{F}\{\mathbf{V}\}$ (line 6). We show it by the following example.

Let us assume that we need 9 multiplications of $\mathcal{F}\{\mathbf{V}\}$. In its binary form, 9 is equal to $\mathcal{B}(9) = 1001$, and let us traverse to each bit of $\mathcal{B}(9)$ one by one (lines 7 – 16). On each bit shift (line 9), we first multiply $\hat{\mathbf{V}}_{sum}$ with $\hat{\mathbf{V}}'$ (line 11) if the bit at the i-th index of $\mathcal{B}(9)$ is equal to 1 (line 10). Then, regardless of the previous computation, after each shift we compute $\hat{\mathbf{V}}'$ as the power of itself (line 13). Following this rule, the computations are:

[1] It has been shown by Milutinovic et al. [22] that the power of two technique reduces computation time also for linear convolution.

Algorithm 3 Fast computation of consecutive summations of a random variable.

```
Data: $X$ – random variable, $n$ – number of necessary convolutions
Result: Sum $X_\Sigma$ equal to $n$ additions of $X$
function fastSum($X, n$):
    $s \leftarrow n \cdot \min(\{x \mid x \in \operatorname{Im} X\}$ // direct application of Proposition 2
    $m_X \leftarrow \max(\{x \mid x \in \operatorname{Im} X\}) - \min(\{x \mid x \in \operatorname{Im} X\}$
    $\mathbf{V} \leftarrow$ compute using Equation (6) and zero-pad until $n \cdot m_X$
    $\hat{\mathbf{V}}_{sum} \leftarrow$ column vector of elements equal to 1, of size $|\mathbf{V}|$
    $\hat{\mathbf{V}}' \leftarrow \mathcal{F}\{\mathbf{V}\}$
    $k, i \leftarrow 0$
    while $k \neq n$ do
        $b \leftarrow \mathcal{B}(n, i)$ // $\mathcal{B}(n, i)$ is a function that checks the binary representation of $n$ and
          returns the value at index $i$
        if $b = 1$ then
            $\hat{\mathbf{V}}_{sum} \leftarrow \hat{\mathbf{V}}_{sum} \odot \hat{\mathbf{V}}'$
        end
        $\hat{\mathbf{V}}' \leftarrow \hat{\mathbf{V}}' \odot \hat{\mathbf{V}}'$
        $k \leftarrow k + b \times 2^i$
        $i \leftarrow i + 1$
    end
    $\mathbf{V} \leftarrow \mathcal{F}^{-1}\left\{\hat{\mathbf{V}}_{sum}\right\}$ // where $v_j$ is the $j$-th element in $\mathbf{V}$
    $X_\Sigma \leftarrow \begin{bmatrix} s+0 & \dots & s+j & \dots & s+n\cdot m_X \\ v_0 & \dots & v_j & \dots & v_{n\cdot m_X} \end{bmatrix}$
return $X_\Sigma$
```

Initial values. $\hat{\mathbf{V}}_{sum} \leftarrow \mathbf{1}$ and $\hat{\mathbf{V}}' \leftarrow \mathcal{F}\{\mathbf{V}\}$
$(i = 0 \text{ and } \mathcal{B}(9, i) = 1)$: $\hat{\mathbf{V}}_{sum} \leftarrow (\hat{\mathbf{V}}_{sum} \odot \hat{\mathbf{V}}') = \mathcal{F}\{\mathbf{V}\}$ and $\hat{\mathbf{V}}' \leftarrow (\hat{\mathbf{V}}')^2 = (\mathcal{F}\{\mathbf{V}\})^2$
$(i = 1 \text{ and } \mathcal{B}(9, i) = 0)$: $\hat{\mathbf{V}}' \leftarrow (\hat{\mathbf{V}}')^2 = (\mathcal{F}\{\mathbf{V}\})^4$
$(i = 2 \text{ and } \mathcal{B}(9, i) = 0)$: $\hat{\mathbf{V}}' \leftarrow (\hat{\mathbf{V}}')^2 = (\mathcal{F}\{\mathbf{V}\})^8$
$(i = 3 \text{ and } \mathcal{B}(9, i) = 1)$: $\hat{\mathbf{V}}_{sum} \leftarrow (\hat{\mathbf{V}}_{sum} \odot \hat{\mathbf{V}}') = (\mathcal{F}\{\mathbf{V}\} \odot (\mathcal{F}\{\mathbf{V}\})^8) = (\mathcal{F}\{\mathbf{V}\})^9$

At the end of this process, $\hat{\mathbf{V}}_{sum}$ is equal to $(\mathcal{F}\{\mathbf{V}\})^9$. The very same principle is used in the algorithm (lines 8 – 16). At the end of the algorithm (lines 17 and 18), it just computes the inverse Fourier Transform of $\hat{\mathbf{V}}_{sum}$, which then characterises X_Σ.

Space-complexity reduction of repetitive computations. Although the time complexity is improved with the above-mentioned methods, there may be the cases where space complexity can still be an issue despite the vector reductions proposed by Improvements 1 and 2. E.g. after k additions of some random variable X, the resulting random variable in the worst-case may have the image that is k times greater than the one of X.

This problem can be further addressed by using the principles implied by the central limit theorem in probability theory. In more details, considering the case when one random variable X is added to itself multiple times, with each new addition the resulting sum tends more towards the normal distributions, even though the X is not normally distributed. Consider for the moment Figure 3, we show the original random variable X, the random variable after three additions of X, and the random variable after 20 additions of X. You can see that by each new summation, the resulting sum resembles more to the normal distribution, although initially it has quite opposite properties compared to it.

Therefore, Algorithm 3 may be improved by applying the down-sampling methods on the inverse Fourier transforms of $\hat{\mathbf{V}}_{sum}$ and $\hat{\mathbf{V}}'$. The proposed down-sampling algorithms are tailored to decrease the vector sizes by safely removing the starting zero tail that contains

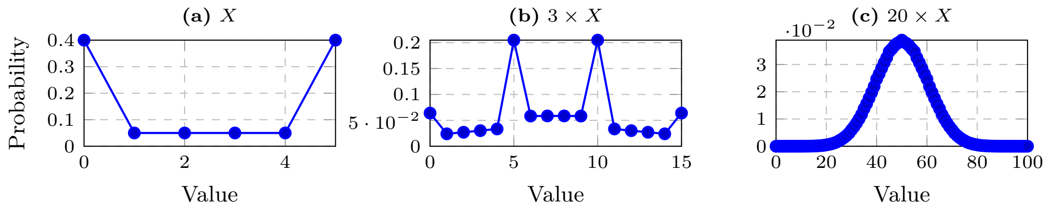


Figure 3 Consecutive additions of the same random variable. The points are connected in order to show the resemblance to the continuous normal distribution.

close-to-zero probabilities. The removed zero interval can then be accounted by Proposition 2. Additionally, the down-sampling method known as domain-quantisation [19] can be an efficient way to further reduce the vector sizes and enable Improvement 1 from Section 4.2.

5 Evaluation

The evaluation is organised in three parts: (i) Evaluation of the down-sampling algorithms, (ii) Evaluation of the convolution computations, and (iii) Computation of the deadline miss probability. The code of the implementation is available (see [17]).

Hardware and software configuration. In all of the experiments, we used a PC with i7 4770k CPU with a frequency of 2.6 GHz, and 32 GB of RAM memory. All the algorithms are implemented in MATLAB, using Advanpix Multiprecision Computing Toolbox [16].

5.1 Evaluation of the down-sampling algorithms

Goal of the evaluation and the evaluated entities. In this evaluation, we compared the two proposed down-sampling algorithms, Optimal (OP), from Section 3, and Linear (LN), from Section 3, with Domain Quantisation (DQ), and Pessimism Reduce (PR), both proposed by Maxim et al. [19]). The later two algorithms are selected since PR introduces the least pessimism in the down-sampled variables among the state-of-the-art algorithms, while DQ is known as the fastest down-sampling algorithm [19]. The comparison between the algorithms is made according to three criteria: (a) Computation time of the algorithms as a function of the size of the initial random variable, (b) Probabilistic expectation, i.e., pessimism introduced upon down-sampling, (c) Probability of exceeding the median. The later is evaluated since the probability of exceeding some value is often used in probabilistic analysis, e.g., deadline miss probability, that is, the probability of exceeding the deadline.

Experiment Setup. In this experiment, we considered $|\mathbf{V}| \in \{200, 400, 600, 800, 1000\}$, and for every cardinality $|\mathbf{V}|$ we sampled 1000 realisations of the random variable X associated with $\mathbf{V}$, using UUniFast algorithm [2]. The experiments analyse the performance of the down-sampling operation to the maximum size of 20 values, for all the 5000 realisations.

Experiment Results. Figure 4(a) shows the average computation time of down-sampling as a function of the cardinality of the initial realisation of the vector $\mathbf{V}$. The results show that the average computation time of OP is impacted the highest (4.7s for the initial size of 1000 values), compared to the other down-sampling approaches. This is due to its cubic time complexity. For the other algorithms, execution time did not exceed 0.01 seconds. According

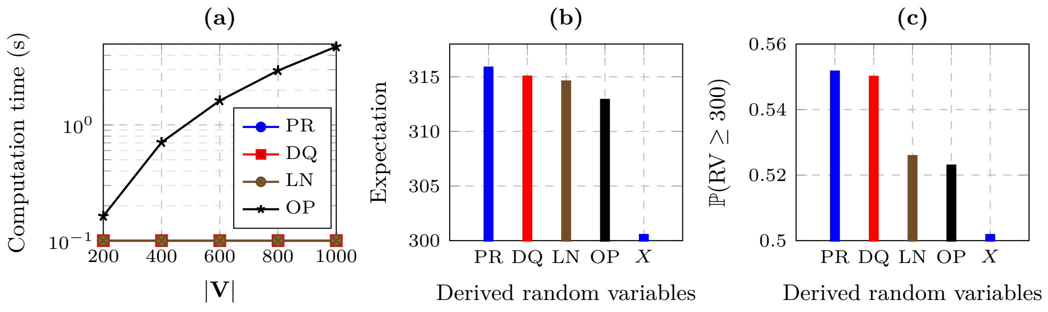


Figure 4 Results of the evaluation on the comparison of down-sampling algorithms.

to MATLAB documentation for *tic toc* functions, execution times which do not exceed the 0.1 seconds are not representative with enough confidence and for this reason are not reported. The average execution time of PR (1μs for the cardinality 1000) seems to be greater than the average execution time of LN and DQ (below 100μs for the cardinality 1000), but this comparison demands for the further, more precise investigation.

Figure 4(b) shows the average probabilistic expectation, for the evaluated algorithms and the expectation of the initial random variable (X in the figure). This figure presents the data only for $|\mathbf{V}| = 600$. OP succeeds to minimise the additional expectation the most, compared to the other algorithms, followed by LN.

Figure 4(c) shows the average probability of exceeding the median, for $|\mathbf{V}| = 600$. The results show that the probability of exceeding 300, is least when using OP, followed by LN.

Discussion. In the above experiments, we showed that OP and LN can introduce less pessimism compared to the state-of-the-art algorithms. This is relevant when the down-sampled variable needs to be used in the probabilistic analysis since the pessimism linearly grows with each new addition to the other variables or itself. The improvements of OP come with the computation cost. Since OP is a parallelisable algorithm, based on dynamic programming, the computation improvements can be further addressed. During the evaluation, we also discovered that each of the evaluated algorithms may derive the best down-sampling with respect to exceedence probability. Therefore, the potential improvement may arise by combining DQ, LN, PR, and OP, which remains for the future work.

5.2 Evaluation of the convolution algorithms

Goal of the evaluation and the evaluated entities. In this evaluation, we compared Algorithm 3, referred in the following as CC (for circular convolution) with the analogous method that uses the power-based addition, for linear convolution, proposed by Milutinović et al. [22], referred as LC (linear convolution). LC is the fastest method in the state-of-the-art for computing the linear convolution of random variables and we implemented it with the linear-convolution function (*conv*) in MATLAB. The comparison criterion in the experiment was the average computation time as a function of the number of convolutions, analogous to the number of summed jobs in the probabilistic schedulability analysis.

Experiment Setup. In this experiment, we generated three random variables, using UUniFast algorithm [2], with size $|\mathbf{V}| \in \{100, 1000, 10000\}$. The results are shown in Figure 5. For each realisation we report the computation time as a function of the number of convolutions

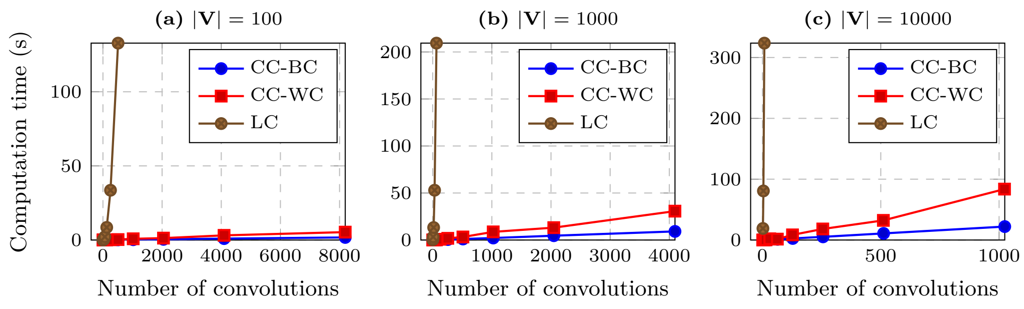


Figure 5 Computation time as a function of number of additions, e.g., number of analysed jobs.

performed. The above-described generation of the probability distribution did not affect the computation times, and therefore we used only one variable per setup. Since Algorithm 3 performs the least number of vector multiplications when the number of convolutions is a power of two, and the largest number of multiplications when the number of convolutions is equal to a power of two minus one, we reported those two lines separately. CC-WC represents the worst-case computation time (worst-case number of vector multiplications) with ticks that are by one less than the power of two, while BC-WC represents the best-case computation time, ticks being powers of two. The two modes of computation do not significantly affect LC, and therefore we reported only the results for ticks equal to powers of two.

Experiment Results. Figure 5(a) shows the case of $|\mathbf{V}| = 100$. We observe that LC takes 132s to compute 512 convolutions, while CC-WC takes only 5s for $8191 = 2^{13} - 1$ convolutions. Due to the smaller number of multiplications, CC-BC takes only 1.7s for performing $8192 = 2^{13}$ convolutions. A similar trend can be observed for $|\mathbf{V}| \in \{1000, 10000\}$ in Figure 5(b)–(c). Finally, it can observed that the size of the $\mathbf{V}$ has a significant impact on the computation time. In fact, looking at CC-WC for $|\mathbf{V}| = 10000$, when the number of convolutions is $1023 = 2^{10} - 1$, the average computation time is 83 seconds, while for CC-BC when the number of convolutions is $1024 = 2^{10}$, the computation time is only 22 seconds.

Discussion. The difference of time complexities of linear and circular convolution can be observed by the derived results. The proposed algorithm, based on the circular convolution and vector reductions defined in the paper, outperforms the linear-convolution-based algorithm in this experiment setup. However, it should be stated that there are cases when LC can provide better computation times compared to CC since the computation time of LC depends on the cardinality of the images of random variables, while the computation time of the circular convolution depends on the difference between the largest and the lowest values within the respective images. Regardless of this, the scaling of the circular convolution is better due to a lower time-complexity class.

5.3 Computation of the deadline miss probability

Goal of the evaluation and the evaluated entities. In order to show the applicability of the proposed computations in the state-of-the-art probabilistic analysis, we implemented the deadline miss probability analysis (DMPA), according to Equation (3) described by von der Brüggen et al. [28], using however the circular convolution for variable additions, instead of the linear convolution (as shown in [18]). In the implementation, we also used Improvement 2 from Section 4.2. Regarding the evaluation, we compared the circular-convolution-based

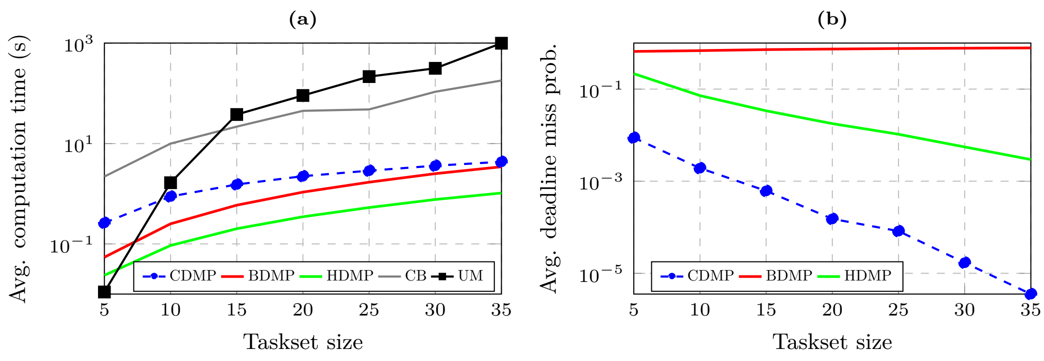


Figure 6 Computation time as a function of a taskset size, measuring deadline miss probability.

approach (CDMP), with the two fastest DMPA approximations, i.e., the Hoeffding bound (HDMP), and the Bernstein bound (BDMP), as defined by von der Brüggen et al. [28]. To the best of the authors' knowledge, these two approximation approaches are the fastest in the state-of-the-art for the given problem. Their drawback is that they tend to over-approximate the deadline-miss probabilities. Compared to the timing values reported by von der Brüggen et al. [28], our implementations for HDMP and BDMP achieve an average speedup factor of 6; for the sake of completeness, we therefore report the average computation time also of the other approaches presented in [28], scaling the timing values according to the speedup factor. The additional methods are: Unify method (UM), proposed by von der Brüggen et al. [28] and the method based on the Chernoff bound (CB), which was proposed by Chen and Chen [6]. UM derives the exact deadline-miss probability, while CB is more accurate and efficient compared to UM, but less efficient approximation compared to HDMP and BDMP.

Experiment Setup. In this experiment, we replicated the setup proposed by von der Brüggen et al. [28]. Thus, 1000 tasksets were generated per each presented point, considering the sizes of $5, 10, \ldots, 35$. For each taskset we generated the utilisations using UUniFast [2], with the taskset utilisation set to 0.7. Task periods were generated according to a log-uniform distribution ranging from 10ms to 1000ms. Normal execution modes were computed by multiplying the utilisation and the period for each generated pair of values. The periods and normal execution mode times were ceiled to be multiples of 50μs. This allowed for the discrete random-variable support while not severely affecting the taskset utilisation given by UUniFast. The implementation of the circular convolution did not include Improvement 1 from the paper, in order to not take the advantage of the experiment setup that considers only two execution time modes. Implicit-deadlines were assumed. Probability of occurrence for the normal execution was set to 0.975, while the probability of occurrence for the abnormal execution mode was set to 0.025. Abnormal execution was set to be two times greater than the normal execution. For more details on the setup, refer to their paper.

Experiment Results. Figure 6(a) shows that CDMP is applicable for computation of the deadline-miss probability. With taskset size of 35, CDMP achieves to compute the deadline miss probability of the lowest priority task, taking 4.3s, while BDMP takes 3.4s, and HDMP around 1s. The projected computation times for CB and UM are still several times larger despite the speedup scaling, although these results may be improved or worsened based on the implementation choices in MATLAB and the underlined hardware. In Figure 6(b), we show the average deadline miss result computed by CDMP, BDMP, and HDMP. Since the result of CDMP is the exact deadline miss probability, we observe that the BDMP estimation becomes more pessimistic with the increase of the taskset size, while the HDMP bound is still significantly worse than the result derived with CDMP (note the log scale).

Discussion. We conclude that the methods proposed in this paper are promising for the deadline miss analysis since the computation time is low, while the results are exact. However, the further empirical investigations should address the impact of the more complex execution time distributions and taskset parameters. For example, using $1\mu s$ as the basic unit of time for random variables may increase the computation time of the method due to necessary vector increase. Also, further investigations should take into account more complex assumptions, e.g. probabilistic executions and inter-arrival times, as proposed by Maxim et al. [18].

6 Related work

As described by Davis and Cucu-Grosjean [10,11], the issues of computational intractability of the linear convolution in the probabilistic schedulability and timing analysis are investigated for years in the research area of real-time systems. There are two main research directions that tried to solve this problem: (i) Down-sampling methods, and (ii) Analytical methods. The goal of the down-sampling methods is to approximate the random variables such that its image size is reduced. Then, the linear convolution can be applied, however introducing over-approximation. There are several works in this domain, e.g., Refaat et al. [25], Diaz et al. [12], Kim et al. [14]. More recent improvement was made by Maxim et al. [19, 20] where several down-sampling algorithms were proposed, among which are *domain quantisation* and *reduced pessimism*. Both were used in the evaluation of this paper. Finally, the most recent improvement was made by Milutinović et al. [22] addressing the cache-miss probability analysis. In their work, several improvements for speeding up the linear convolution were proposed, using parallel computations and power operations, while they also pointed that the use of circular convolution may be relevant for addressing the tractability issues. Contrary to the above methods, Chen and Chen [6] proposed an analytical approach for computing the deadline-miss probability, which is based on the *Chernoff bounds*. Following this line of research, von der Brüggen et al. [27, 28] proposed several algorithms for approximating the deadline miss probability (based on the Hoeffding and the Bernstein inequalities). Next to those, they also proposed two exact methods: *Pruning*, which is a multinomial-based approach combined with the pruning techniques, and *Unify*, which is a combination of *Pruning* with the approach based on the union of equivalence classes. More recently Chen et al. [7] proposed the improvement for the Chernoff bounds approximation, which is solved by considering an equivalent convex optimisation problem. This improvement reduces the computation time of deriving the approximation, while it also improves its accuracy.

7 Conclusions

In this paper, we addressed two problems that consider random variables and their use in the analysis of probabilistic real-time systems. The first problem considered the space reduction, i.e., the down-sampling of a random variable. This is the problem of approximation, such that the distribution of the random variable is preserved while its set of values is reduced. Such process in the probabilistic analysis often introduces pessimism, that then propagates towards the final results of the probabilistic response-time and similar types of analysis. We proposed an optimal algorithm for down-sampling, which minimises upon probabilistic expectation, i.e., pessimism introduced upon down-sampling.

The second addressed problem considers the efficiency of computing the convolution between random variables. This problem for years limited the applicability of many existing and possibly future work in the domain of probabilistic analysis of real-time systems. In this paper, we showed how the circular convolution can be used to address this problem, reducing the time complexity of a single discrete convolution from $O(n^2)$ to $O(n\log(n))$. Using the circular convolution with the vector reductions proposed in the paper, we showed in the evaluation that the proposed approach shows promising results with respect to its applicability in the existing problems of probabilistic analysis.

References

1 George B. Arfken and Hans J. Weber. Mathematical methods for physicists, 1999.

2 Enrico Bini and Giorgio C Buttazzo. Measuring the performance of schedulability tests. *Real-Time Syst.*, 30(1-2):129–154, 2005.

3 Jonathan M. Blackledge. *Digital image processing: mathematical and computational methods.* Elsevier, 2005.

4 Sergey Bozhko and Björn B Brandenburg. Abstract response-time analysis: A formal foundation for the busy-window principle. In *Euromicro Conf. Real-Time Syst. (ECRTS 2020)*, 2020.

5 Ronald Newbold Bracewell. *The Fourier transform and its applications.* McGraw-Hill, 1999.

6 Kuan-Hsun Chen and Jian-Jia Chen. Probabilistic schedulability tests for uniprocessor fixed-priority scheduling under soft errors. In *IEEE Int. Symp. Industrial Emb. Syst. (SIES)*, pages 1–8, 2017.

7 Kuan-Hsun Chen, Niklas Ueter, Georg von der Brüggen, and Jian-Jia Chen. Efficient computation of deadline-miss probability and potential pitfalls. In *Design, Automation & Test in Europe Conf. & Exhibition (DATE)*, pages 896–901, 2019.

8 Kuan-Hsun Chen, Georg von der Brüggen, and Jian-Jia Chen. Analysis of deadline miss rates for uniprocessor fixed-priority scheduling. In *IEEE Int. Conf. Emb. and Real-Time Computing Syst. and Applications (RTCSA)*, pages 168–178, 2018.

9 James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

10 Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic schedulability analysis techniques for real-time systems. *LITES: Leibniz Trans. Emb. Syst.*, pages 1–53, 2019.

11 Robert Ian Davis and Liliana Cucu-Grosjean. A survey of probabilistic timing analysis techniques for real-time systems. *LITES: Leibniz Transactions on Embedded Systems*, pages 1–60, 2019.

12 Jose Luis Diaz, Jose Maria Lopez, Manuel Garcia, Antonio M Campos, Kanghee Kim, and Lucia Lo Bello. Pessimism in the stochastic analysis of real-time systems: Concept and applications. In *IEEE Int. Real-Time Syst. Symp. (RTSS)*, pages 197–207, 2004.

13 Geoffrey Grimmett and Dominic Welsh. *Probability: an introduction.* Oxford U. Press, 2014.

14 Kanghee Kim, Jose Luis Diaz, Lucia Lo Bello, José María López, Chang-Gun Lee, and Sang Lyul Min. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *IEEE Transactions on Computers*, 54(11):1460–1466, 2005.

15 Charan Langton and Victor Levin. *The Intuitive Guide to Fourier Analysis and Spectral Estimation.* Mountcastle Company, 2017.

16 Advanpix LLC. Multiprecision computing toolbox for MATLAB. URL: `http://www.advanpix.com/`.

17 Filip Marković, Alessandro Vittorio Papadopoulos, and Thomas Nolte. Artifact-evaluation—on-the-convolution-efficiency, 2021. URL: `https://github.com/Aeoliphile/Artifact-Evaluation---On-the-convolution-efficiency`.

18 Dorin Maxim and Liliana Cucu-Grosjean. Response time analysis for fixed-priority tasks with multiple probabilistic parameters. In *2013 IEEE 34th Real-Time Systems Symposium*, pages 224–235. IEEE, 2013.

19 Dorin Maxim, Mike Houston, Luca Santinelli, Guillem Bernat, Robert I Davis, and Liliana Cucu-Grosjean. Re-sampling for statistical timing analysis of real-time systems. In *Int. Conf. Real-Time and Network Syst. (RTNS)*, pages 111–120, 2012.

20 Dorin Maxim, Luca Santinelli, and Liliana Cucu-Grosjean. Improved sampling for statistical timing analysis of real-time systems. *Int. Conf. Real-Time and Network Syst. (RTNS)*, pages 17–20, 2010.

21 Stephen McGovern. MATLAB Central File Exchange, Fast Convolution. `https://www.mathworks.com/matlabcentral/fileexchange/5110-fast-convolution`. Accessed: 2021-01.

22 Suzana Milutinović, Jaume Abella, Damien Hardy, Eduardo Quiñones, Isabelle Puaut, and Francisco J Cazorla. Speeding up static probabilistic timing analysis. In *Int. Conf. Architecture of Computing Syst. (ARCS)*, pages 236–247, 2015.

23 Alan V Oppenheim, John R Buck, and Ronald W Schafer. *Discrete-time signal processing. Vol. 2.* Upper Saddle River, NJ: Prentice Hall, 2001.

24 Athanasios Papoulis. The fourier integral and its applications. *McCraw-Hill*, 1962.

25 Khaled S Refaat and Pierre-Emmanuel Hladik. Efficient stochastic analysis of real-time systems via random sampling. In *Euromicro Conf. Real-Time Syst. (ECRTS)*, pages 175–183, 2010.

26 Moshe Shaked and J. George Shanthikumar, editors. *Univariate Stochastic Orders*, pages 3–79. Springer New York, New York, NY, 2007.

27 Georg von der Brüggen. *Realistic Scheduling Models and Analyses for Advanced Real-Time Embedded Systems.* PhD thesis, TU Dortmund (Germany), 2019.

28 Georg von der Brüggen, Nico Piatkowski, Kuan-Hsun Chen, Jian-Jia Chen, and Katharina Morik. Efficiently approximating the probability of deadline misses in real-time systems. In *Euromicro Conf. Real-Time Syst. (ECRTS)*, 2018.