



Statistical static timing analysis via modern optimization lens: I. Histogram-based approach

Adam Bosák¹ · Dmytro Mishagli² · Jakub Mareček³

Received: 8 November 2022 / Revised: 27 August 2023 / Accepted: 27 August 2023
© The Author(s) 2023

Abstract

Statistical Static Timing Analysis (SSTA) is studied from the point of view of mathematical optimization. We present two formulations of the problem of finding the critical path delay distribution that were not known before: (i) a formulation of the SSTA problem using Binary–Integer Programming and (ii) a practical formulation using Geometric Programming. For simplicity, we use histogram approximation of the distributions. Scalability of the approaches is studied and possible generalizations are discussed.

Keywords Binary–Integer Programming · Geometric Programming · Statistical Static Timing Analysis

1 Introduction

Integrated Circuits (ICs) must work at expected frequencies with respect to the timing constraints specified in their designs, which is checked by Computer-Aided Design (CAD) tools. The key challenge is to satisfy these constraints in an optimal way, so that the area taken by the designs on a chip and the power consumption are minimized. This leads to a particular class of optimization problems. The development of the

✉ Jakub Mareček
jakub.marecek@fel.cvut.cz

Adam Bosák
bosadam@seznam.cz

Dmytro Mishagli
dmytro.mishagli@ucd.ie

¹ Technical University of Denmark, Kongens Lyngby, Denmark

² University College Dublin, Belfield, Ireland

³ Czech Technical University in Prague, Prague, Czech Republic

techniques used for design verification forms a separate field of study, the Timing Analysis (Sapatnekar 2004).

With the decrease of features sizes, the impact of variations that occur in manufacturing processes, the process variations, increases. These variations lead to uncertainties in the parameters of the transistors and interconnects, which affect the delays and hence the overall performance of a circuit. For example, the performance of the same designs can differ from chip to chip (intradie or global variations). At the same time, a single design can have variation of delays in different parts of a die (inter-die or local variations). The most reliable way to take these variations into account in order to predict the yield (i.e., the fraction of correctly functional chips among all fabricated) is to run Monte Carlo (MC) simulations. For modern Very Large Scale Integration (VLSI) designs, this is very expensive, which again increases the cost of chips.

A less computationally expensive approach is to use a Static Timing Analysis (STA), which is still the most common way to take into account systematic (global) process variations (Sapatnekar 2004; Vygen 2006; Bhasker and Chadha 2009). This approach treats variations as single and determined values (so-called *corner* values). The delay values computed in such a way are too pessimistic (Blaauw et al. 2008), which results in an increase in the cost of chips when these delays are mitigated (Visweswariah 2003). An additional challenge is that variations have substantial non-Gaussian behaviour and are often strongly correlated. In modern ultra-VLSI circuits (5 nm and below), the impact of random correlated processes and fluctuations has become even more important. Thus, an alternative approach has been developed.

Statistical Static Timing Analysis (SSTA) addresses randomness in a natural way, treating delays in a system as Random Variables (RVs) from the very beginning. The analysis then allows us to determine the mean value of the delay across selected paths. The maximum delay corresponds to the critical path. Current industrial realizations of SSTA allow one to determine moments of delay distributions and/or their quantiles (Chang and Sapatnekar 2003, 2005; Chang et al. 2005). In principle, SSTA can give a delay distribution of the whole circuit. This makes SSTA comparable to MC simulations in terms of accuracy. At the same time, SSTA algorithms are much less resourceful than MC but have higher computational complexity than deterministic STA.

This work is motivated by Freeley et al. (2018); Mishaghi and Blokhina (2020), where an approach was proposed to deal with non-Gaussian distributions of gates' delays without loss of information while keeping complexity low enough. The authors based their method on the exact solution to the problem of finding a distribution of a logic gate delay assuming that all distributions are Gaussian. The resulting non-Gaussian distribution is then decomposed into a mixture of Radial Basis Functions with fixed shape parameters and locations, but unknown mixing coefficients (the weights). Such a mixture, which is referred to as *Gaussian comb*, can be obtained by means of a Linear Programming. This is the main advantage of the proposed model.

The aim of this paper is to investigate a classical SSTA problem from the mathematical optimization point of view, which is, to the best of our knowledge, done for the first time. For simplicity, we will represent actual distributions with histograms, which inevitably introduces accuracy drop. We make this choice deliberately, as our focus is on proofing the concept rather than developing an accurate approach. We give two

formulations of the SSTA problem as an optimisation problem *via* (i) Binary–Integer Programming and (ii) Geometric Programming. Finding efficient solutions will be the subject of a separate study where we combine the approach presented in this work with the Gaussian comb model.

The paper is organized as follows. In Sect. 2, we give necessary mathematical preliminaries, terminology, and discuss related work. Section 3 discusses standard and straightforward (i.e., non-optimization) approaches to SSTA and gives the statement of the problem. We give our solutions *via* optimization techniques in Sects. 4 and 5. Section 4 discusses formal mathematical solution by means of Binary Integer Programming. Section 5 shows that the scalability can be improved by utilizing Geometric Programming. We then conclude the paper in Sect. 6 with overall discussion of the obtained results.

2 Background and related work

In this Section, we start from some mathematical preliminaries, followed by introducing the terminology, and finally briefly describe the key results in the Statistical Static Timing Analysis.

2.1 Mathematical preliminaries

Here we will present only necessary definitions and refer to Boyd et al. (2007) for a very accessible overview of Geometric Programming. For further details, we refer the reader to the references to the literature cited there.

Monomial is a function: $\mathbb{R}_{++}^n \rightarrow \mathbb{R}_{++}$ ¹

$$f(x_1, x_2, \dots, x_n) = cx_1^{a_1} \dots x_n^{a_n}, \tag{1}$$

where $c > 0$ and $a_i \in \mathbb{R}$. We call c the coefficient of the monomial and a_i exponents of the monomial. We refer to a sum of monomials as a posynomial, that is, a function of the form

$$f(x_1, x_2, \dots, x_n) = \sum_{k=1}^K c_k x_1^{a_{1k}} \dots x_n^{a_{nk}}. \tag{2}$$

It is easy to see from the definitions that for the set of all monomials A and for the set of all posynomials B , it holds $A \subseteq B$. One should also note that posynomials are closed under addition, multiplication, positive scaling, and results in a posynomial, when divided by a monomial.

A Geometric Program (GP) is an optimization problem of the form

$$\begin{aligned} &\text{minimize} && f_0(\mathbf{x}) \\ &\text{subject to} && f_i(\mathbf{x}) \leq 1, \quad i = 1, \dots, n \\ &&& g_i(\mathbf{x}) = 1, \quad i = 1, \dots, p, \end{aligned} \tag{3}$$

¹ The domain of the monomials is the non-negative quadrant of \mathbb{R}^n . We assume that the optimum values cannot be zero, and therefore the domain is of the form \mathbb{R}_{++}^n .

where x_i are optimization variables, f_i are posynomial functions, and g_i are monomials. We call (3) a GP program in a standard form.

An Integer Linear Programming (ILP) problem is written in general form as follows (see, e.g., Wolsey and Nemhauser (1999)):

$$\begin{aligned} & \text{maximize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{Ax} \leq \mathbf{b}, \\ & && \mathbf{x} \geq \mathbf{0}, \\ & && \mathbf{x} \in \mathbb{Z}^n. \end{aligned} \quad (4)$$

If the domain polyhedron is intersected with the hypercube $\{0, 1\}^n$, we talk about Binary–Integer Programming (BIP) problems. This can be specified using constraints

$$\begin{aligned} & \mathbf{0} \leq \mathbf{x} \leq \mathbf{1}, \\ & \mathbf{x} \in \mathbb{Z}^n. \end{aligned} \quad (5)$$

2.2 Definitions

Throughout the paper, we will use the following commonly accepted terminology (Blaauw et al. 2008). A logic circuit can be represented as a timing graph $G(E, V)$, where the *graph* and its *paths* are defined as follows.

Definition 1 A timing graph $G(E, V)$ is an acyclic directed graph, where E and V are the sets of edges and vertices, respectively. The vertices correspond to logic gates of a circuit. The timing graph always has one source and one sink. The edges are characterised by weights d_i that describe delays. The timing graph is called a *statistical timing graph* within SSTA when the edges of the graph are described by RVs.

The task then is to determine the critical (longest) path.

Definition 2 Let $p_i (i = 1, \dots, N)$ be a path of ordered edges from the source to the sink in a timing graph G and let D_i be the path length of p_i . Then $D_{\max} = \max(D_1, \dots, D_N)$ is called the *SSTA problem of a circuit*.

Since the gates have internal structure presented by corresponding combination of transistors, this results in a characteristic time needed for the gates to operate. This is one of the sources of delays in a circuit. Due to delays, input signals can have different arrival times, and therefore, the delay of a gate is determined by the maximum of input delays. On the other hand, the operation time of a gate can have a significant impact on the circuit delay, in addition to the arrival times. In such a case, the delay of a gate itself should be added to the result of the max function:

$$d_{\text{gate}} = \max(d_1, d_2) + d_0 + d_{\text{int}} + \dots, \quad (6)$$

where d_1, d_2 are delays in input signals, d_0 is a gate delay (due to its operation time) and d_{int} is an interconnect delay.

The calculation is straightforward in the case of a deterministic timing analysis, but is not the case when uncertainty arises. As we have already mentioned, within the SSTA, the arrival and gate operation times are described by RVs given by the corresponding distributions. Therefore, the delay (6) can be written as

$$\zeta_{\text{gate}} = \max(\xi_1, \xi_2) + \xi_0 + \xi_{\text{int}} + \dots, \quad (7)$$

where ξ_1 and ξ_2 are RVs that describe the arrival times of input signals, ξ_0 and ξ_{int} are RVs related to the gate operation time and the interconnect delay respectively. The whole gate delay, d_{gate} , is now an RV itself and is indicated by ζ_{gate} . In principle, ξ_0 and ξ_{int} can be combined in a single RV, thus, the latter will be omitted in future discussion.

Therefore, as we can see from (7), two operations fully describe delay propagation at the gate level: (i) the maximum of delays entering a gate and (ii) the summation of the latter with the delay of the gate. These operations are often called *atomic operations* of SSTA (see, e.g., works by Cheng et al. 2007, 2012). In the language of distributions, Eq. (7) gives a *convolution* of probability density functions of the RVs $\max(\xi_1, \xi_2)$ and ξ_0 . In this work, we consider a histogram approximation to the problem, which will be discussed in the next sections.

2.3 Related work

There were excellent reviews of the work done in the early stage of the SSTA era, 2001–2009, namely by Blaauw et al. (2008) and Forzan and Pandini (2009). A good overview is also conducted by Beece et al. (2014), where a transistor sizing problem was addressed by means of optimization techniques. We shall summarise key ideas of the SSTA research in this subsection.

The research at that stage was based on variants of the idea first presented by Clark (1961) within the block-based approach. The idea is that actual distributions can be approximated by Gaussians by matching the first two moments (mean and variance). Thus, in Chang and Sapatnekar (2003, 2005) Clark's algorithms were accompanied by handling spatial correlations using principal component analysis (PCA). To propagate a delay through the timing graph, the linear *canonical model* of a delay was proposed (Visweswariah et al. 2004; Chang et al. 2005; Visweswariah et al. 2006). The delay is described as a linear function of parameter variations:

$$D = a_0 + \sum_{i=1}^n a_i \Delta X_i + a_{n+1} \Delta R, \quad (8)$$

where a_0 is the mean or nominal value, ΔX_i represents the variation of n global sources of variation X_i from their nominal values, a_i are the sensitivities to each of the RVs, and ΔR is the variation of the independent RV, R . Then, the mean and variance of a delay were represented using a concept of *tightness probability* (or *binding probability*

Algorithm 1: General SSTA algorithm**Data:** Distributions of delays for N gates and for the input signals**Result:** A distribution of the max delay of a circuit

```

1 for  $i \leftarrow 1, \dots, N$  do
2    $M \leftarrow$  max of arrival times;
3    $C \leftarrow$  convolution of the  $i^{\text{th}}$  gate and  $M$ ;
4   propagate  $C$  further as input PDF;
5  $D \leftarrow$  max of output distributions;

```

in Jess et al. (2003, 2006)) $T_A = P(A > B)$, which is the probability that the arrival time A is greater than B . The linear approximation for $\max(A, B)$ was proposed.

Also, various extensions to this approach were proposed mainly based on adding non-linear terms to (8). For example, the quadratic term was introduced in Zhan et al. (2005, ?, 2006). Chang et al. (2005) proposed to use numerically computed tables to describe the non-linear part of the canonical form. Khandelwal and Srivastava (2005) considered gate delays and arrival times using their Taylor-series expansion. The paper (Ramprasath et al. 2016) discusses another modification of the canonical form (8), based on the addition of the quadratic term and using skew-normal distributions. The correlations were considered in Chang and Sapatnekar (2005), as we mentioned above, within the PCA method. Later, in Singh and Sapatnekar (2008), it was proposed to transform the set of correlated non-Gaussian variables *via* an independent component analysis (ICA) into a non-correlated set. The described canonical delay model suffers from a big disadvantage: it requires approximation of the maximum operation, which is a source of errors that we want to mitigate.

More broadly, optimisation problems appear in various aspects of CAD for VLSI (Korte and Vygen 2008; Brenner and Vygen 2009; Held et al. 2011). For example, the gate sizing problem has received attention in the community for more than 30 years (Berkelaar and Jess 1990; Sapatnekar et al. 1993; Jacobs and Berkelaar 2000; Rakai and Farshidi 2015). One should point out the GP formulations of this optimization problem (Boyd et al. 2005, 2007; Joshi and Boyd 2008; Naidu 2021), as the latter plays a significant role in the present study, but we shall not discuss these works in detail, as the gate sizing is out of scope in the present paper. One should note that in these papers, as well as in the above mentioned work (Beece et al. 2014), timing analysis was performed using the canonical model of a delay.

3 Statistical static timing analysis: setting up the problem

This Section revisits the problem of calculating the maximum delay in the SSTA histogram approximation, as captured in Algorithm 1. In particular, we will present the exact computation of the maximum and the convolution, without claiming novelty of the presented material.

Throughout this paper, distributions are represented by the histogram approximation, where we assume that all histograms share edges of all bins. This assumption

Algorithm 2: Maximum of two histograms

Data: Number of bins n , two vectors x, y of histogram values

Result: A new vector $z = \max(x, y)$

```

1  $z \leftarrow \mathbf{0}$ ;
2 for  $i \leftarrow 1, \dots, n$  do
3   for  $k \leftarrow 1, \dots, i$  do
4      $z[i] \leftarrow z[i] + x[i] \cdot y[k]$ ;           /* first term in (11) */
5     if  $i \neq k$  then
6        $z[i] \leftarrow z[i] + y[i] \cdot x[k]$ ;           /* second term in (11) */

```

is made for the sake of clarity and can be removed at the cost of a somewhat more complicated notation.

Let us find the histogram approximation of (i) a distribution of the maximum ζ of two independent random variables η, ξ , and (ii) a convolution of two histograms. We assume a set of bins $B = \{0, 1, \dots, n - 1\}$, and that the histogram samples η_i and ξ_i take the value in the interval $[a, b], \forall i$. Given the edges interval $[a, b]$, we partition \mathbb{R} into n intervals of size $|b - a|/n$ with points e_1, e_2, \dots, e_{n+1} such that e_i and e_{i+1} are the start and end of the i^{th} interval correspondingly. The midpoints of the intervals m_1, m_2, \dots, m_N are also given.

3.1 Maximum operation

The maximum of two RVs, $\zeta = \max(\eta, \xi)$, which is an RV itself, is defined as

$$\zeta = \begin{cases} \eta, & \text{if } \eta \geq \xi, \\ \xi, & \text{if } \eta < \xi. \end{cases} \tag{9}$$

Using the law of total probability and taking into account the independence of ξ and η , the probability P of a realization $\zeta = z$ can be written as

$$P(\zeta = z) = P(\eta = z) \cdot P(\xi \leq z) + P(\xi = z) \cdot P(\eta < z). \tag{10}$$

It is easy to see that the discrete random variable formulation and histogram estimations h_ζ, h_η , and h_ξ for the RVs ζ, η , and ξ are expressed as follows:

$$h_\zeta[i] = h_\eta[i] \cdot \sum_{k=1}^i h_\xi[k] + h_\xi[i] \cdot \sum_{k=1}^{i-1} h_\eta[k]. \tag{11}$$

Note that the upper bound of the second sum is $i - 1$. This is due to a strict inequality, $\eta < \xi$, in (9). This is summarized as the Algorithm 2.

3.2 Convolution operation

The convolution of two discrete-valued functions, f and g , is defined as:

$$(f * g)[z] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[z - k]. \quad (12)$$

Time complexity of the naïve implementation of the convolution is $\mathcal{O}(N^2)$, which is can be seen in Algorithm 3. The formula (12) implies that the values of the edges must be changed. The value of the first edge has to be added to all other edges. This can be done in many ways and is discussed in the following.

One can add the first value to all edges and unite them during the SSTA algorithm when the edges of the second histogram differ. The receipt is simple: find a new array of edges \mathbf{e} and modify the PDFs of histogram approximations f_α, f_β with the new changed edges. The array of edges of f_α is given as \mathbf{e}^α , similarly \mathbf{e}^β denotes the array of edges of f_β .

Similar to the `rv_histogram()` method of the `scipy.stats` library, one can find a distribution function (PDF) that fits the given histogram as follows. Let us say that F is the fitted cumulative distribution function (CDF) of the f_α . Then PDF of the realization $z \in n_i$ of the new histogram $f_{\alpha'}$ with the desired edges \mathbf{e} is

$$f_{\alpha'}(z) = F(e_{i+1}) - F(e_i), \quad (13)$$

or, recalling a definition of CDF,

$$f_{\alpha'}(z) = \int_{e_i}^{e_{i+1}} f_\alpha(x) dx. \quad (14)$$

The same would be done for $f_{\beta'}$. The solution (14) gives more precise results and bypasses the problem of fitting functions, which is relatively computationally demanding, all at the cost of a slightly longer code. The exact integration can be performed in $\mathcal{O}(N)$ for the whole histogram (Fig. 1).

When one looks at the SSTA Algorithm 1, a problem with such a union of edges is evident. After convolutions of the input gates, every time a maximum and then convolution are computed, the edges will differ and have to be united. Taking into account the two inputs for each gate, the function (13) or (14) is called twice per a gate. Moreover, more problems are to come when trying to solve a convolution optimization problem.

A different and more straightforward solution is presented in the next subsection.

3.3 Convolution with shifting a histogram

A second solution to the problem of adding a value to the edges is a simple shifting. We can shift the whole histogram to the left or right by the number of bins determined by a value that is to be added to all edges divided by the length of the bins and floored.

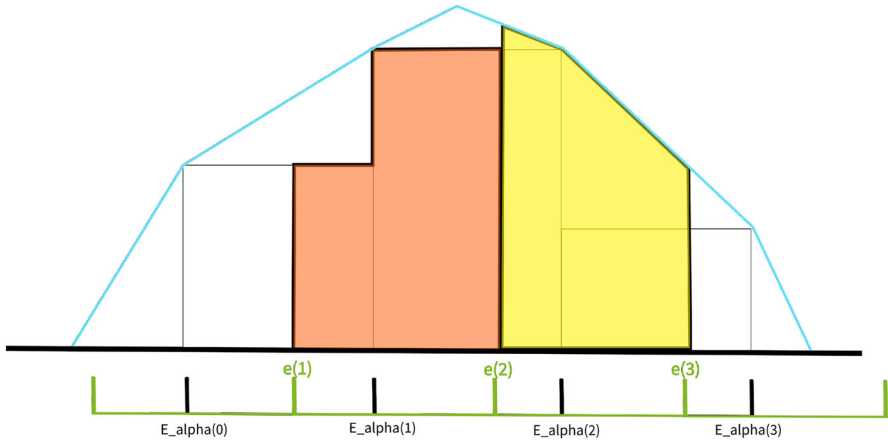


Fig. 1 Sketch demonstrating integration over the fitted function (13) (yellow surface) and the exact integration (14) (orange shaded area to the left). Blue function represents the fitted PDF. (Color figure online)

Shifting the value of a bin by such a number simulates the addition of the first value to all edges. We assume n bins, set of bins $B = \{0, 1, \dots, n - 1\}$ the identical edges of the histograms is given as $\mathbf{e} \in \mathbb{R}^{(n+1) \times 1}$, two RVs α, β ; their convolution ζ , its shifted version ζ' ; and their histogram approximations $h_\alpha, h_\beta, h_\zeta, h_{\zeta'}$. The shift s can be computed as

$$s = \left\lfloor \frac{|e_0|}{e_1 - e_0} \right\rfloor, \tag{15}$$

where $\lfloor \cdot \rfloor$ denotes the flooring operation. In the case of $e_0 > 0$, the new changed histogram $h_{\zeta'}$ at point $x \in B$ will look like

$$h_{\zeta'}[x + s] = h_\zeta[x] \tag{16}$$

In the case of $e_0 < 0$, the shift is similar:

$$h_{\zeta'}[x] = h_\zeta[x + s] \tag{17}$$

When shifting to the right, there are s unoccupied positions on the left. These are nullified. Similarly, done when shifting to the left. Having the starting interval set correctly, this does not have any effect on precision as the starting and ending bins should always be zero. If the interval is small, the accuracy increases, since the bins can encode more information in a smaller interval. However, if it is too small, then we cut some information by this shift. Furthermore, the more bins we add, the more precise this shift will be.

The shifting method gives exactly precise solutions as the one with union. It is straightforward to implement, can be done in linear time, and can be used nearly without any change in the optimization problem. Therefore, this method is used better than the union method.

Algorithm 3: Convolution

Data: Number of bins n , two vectors x, y of histogram values, edges array e
Result: Convolution of two histograms in c

```

1  $c \leftarrow \mathbf{0}$ ;
  // perform convolution
2 for  $z \leftarrow 1, \dots, n$  do
3   for  $k \leftarrow 1, \dots, z$  do
4      $c[z] \leftarrow c[z] + x[k] \cdot y[z - k]$ ;

  // shift histogram
5  $s \leftarrow \text{floor}(\text{abs}(e[0]) / (e[1] - e[0]))$ 
6 if  $e[0] > 0$  then
7    $c[s:] \leftarrow c[: -s]$ ;
8    $c[: s] \leftarrow 0$ ;
9 if  $e[0] < 0$  then
10   $c[: -s] \leftarrow c[s:]$ ;
11   $c[-s:] \leftarrow 0$ ;

```

In summary, we have reviewed two key (atomic) operations of SSTA algorithms, maximum and convolution, and the way how these operations can be performed for histograms. In the following sections, we will give a formulation of the SSTA as (i) a Binary Integer Programming problem and (ii) a Geometric Programming problem. These formulations given for histograms constitute the original contribution of this work.

4 SSTA via Binary–Integer programming

Here, we formulate the SSTA algorithm as a Binary–Integer Programming (BIP) problem. For this purpose, we will introduce a unary encoding of counts in binary variables.² Then, we will show how to perform the atomic operations (max and convolution) on histograms *via* BIP. Finally, we will compare the approach against Monte Carlo simulations and discuss its scalability.

4.1 Statement of the problem

To write the SSTA Problem of a Circuit, $D_{\max} = \max(D_1, \dots, D_N)$, as a BIP problem (4)–(5), one should (i) propose a risk measure for the objective function $\mathbf{c}^T \mathbf{x}$, and (ii) formulate corresponding constraints. Let us discuss the constraints first.

From the Algorithms 2 and 3, one can see that a multiplication of two non-negative real numbers occurs in both of them. By utilizing multiplication naively, we obtain a bi-linear function, which is not convex jointly in both arguments. This cannot be used as a constraint or as an objective function. One solution could be to use McCormick

² This may seem confusing, at first, but we do utilize *binary* decision variables to store *unary*-encoded counts. The *unary* encoding allows for an easier formulation of the corresponding constraints than in the case of *binary* encoding in *binary* variables, which is also possible (Vielma and Nemhauser 2011).

envelopes. This requires setting the lower and upper bounds of the factors. The only way to compute these bounds is by the exact computation of the problem using the methods shown in Sect. 3. Another option is to formulate the problem in unary notation, which is the subject of the present Section.

Below, we will discuss how the atomic operations of SSTA can be performed on histograms in unary encoding. It will be shown that this leads to the corresponding mixed-integer linear programs.

4.2 Atomic operations on histograms in unary representation

As we have discussed above, multiplication is the key operation for both max and convolution. This Section discusses the multiplication in unary representation.

4.2.1 Multiplication

Let us first make a note on vectorization of the multiplication operation. Consider two natural numbers, α and β . By definition, the multiplication of two numbers is equal to the repeated addition:

$$\alpha \cdot \beta = \underbrace{\beta + \beta + \dots + \beta}_{\alpha \text{ times}}. \tag{18a}$$

It is easy to see that in unary notation it can be represented by matrix-vector multiplication. Indeed, having written α and β as column vectors, \mathbf{a} and \mathbf{b} , of sizes $\alpha \times 1$ and $\beta \times 1$, correspondingly, we obtain

$$\alpha \cdot \beta = \mathbf{1}_{1 \times \alpha}^T \begin{bmatrix} \mathbf{a} & \mathbf{a} & \dots & \mathbf{a} \end{bmatrix}_{\alpha \times \beta} \mathbf{b}_{\beta \times 1} \tag{18b}$$

This representation allows performing multiplication of numbers written in unary encoding efficiently. Now, we shall discuss multiplication of *binary variables* by means of integer programming.

Consider two binary variables: $x, y \in \{0, 1\}$. Multiplication of the variables via BIP requires introduction of an auxiliary variable, $s \in \{0, 1\}$, and setting corresponding constraints. The constraints establish obvious relationship between this variable, s , and the multiplicands, x and y :

$$\left. \begin{aligned} s &\leq x, \\ s &\leq y, \\ s &\geq x + y - 1. \end{aligned} \right\} \tag{19}$$

The process of formulating the constraints (19) for the convolution of two histograms in binary form is summarized as Algorithm 4. Note that the two outer loops, over z and k , correspond to those from Algorithm 3, and the inner loops, over i and j , are due to unary encoding of the counts. However, these inner loops can be removed if the computation performed via efficient vector operations as shown in (18b).

Algorithm 4: Generation of BIP constraints for convolution operation

Data: Number of bins n , number of binary variables m , two histograms in unary encoding, \mathbf{H}_X and \mathbf{H}_Y , of size $n \times m$

Result: 1-D array \mathbf{u} of size n with auxiliary variables s and a list with the corresponding constraints

```

1  $\mathbf{u} \leftarrow \mathbf{0}$ 
  // for all bins of a histogram  $\mathbf{H}_X$ 
2 for  $z \leftarrow 1, \dots, n$  do
  // for all bins of a histogram  $\mathbf{H}_Y$ 
3   for  $k \leftarrow 1, \dots, z$  do
    // for all unary digits of the bin of the histogram  $\mathbf{H}_X$ 
4     for  $i \leftarrow 1, \dots, m$  do
      // for all unary digits of the bin of the histogram  $\mathbf{H}_Y$ 
5       for  $j \leftarrow 1, \dots, m$  do
6         initialize a variable,  $s$ 
7          $\mathbf{u}[z] \leftarrow \mathbf{u}[z] + s$ 
8          $x \leftarrow \mathbf{H}_X[k, i]$ 
9          $y \leftarrow \mathbf{H}_Y[z - k, j]$ 
10        add unary constraints (19) linking  $s$ ,  $x$ , and  $y$  for the specific pair  $(i, j)$  of unary
        digits in the histograms  $\mathbf{H}_X, \mathbf{H}_Y$ 

```

The BIP constraints for the max operation can be introduced in a similar manner and, hence, not discussed here. This procedure will repeat two loops from Algorithm 2 and will have two internal loops as in Algorithm 4. Therefore, further in this Section we will discuss only convolution operation, but we want to note that the same argumentation can be carried out for the maximum of two histograms.

4.2.2 Forming a histogram in the unary encoding

Consider two histograms in unary encoding, \mathbf{H}_X and \mathbf{H}_Y , of size $n \times m$, where n is the number of bins and m is the number of binary variables in a bin. For these histograms, the Algorithm 4 returns an array \mathbf{u} with auxiliary variables s given as variables. The values of these variables shall be obtained during the solution of the BIP problem.

One can see that the length of the array \mathbf{u} equals the number of bins, n . Each element of the array \mathbf{u} contains a sum of the auxiliary variables (see Algorithm 4, lines 6–7). It is easy to see the meaning of the each element in \mathbf{u} (and, hence, of the each sum of s variables). They correspond to the values of bins in a histogram \mathbf{H}_{XY} , which is the convolution of \mathbf{H}_X and \mathbf{H}_Y .

In order to map the elements of \mathbf{u} onto the bin values of the histogram \mathbf{H}_{XY} , we (i) generate the matrix $\mathbf{H}_{XY} \in \{0, 1\}^{n \times m}$ with *new* variables, and (ii) supplement these variables with the following constraints

$$\left. \begin{array}{l} \mathbf{1}^T \mathbf{H}_{XY}[1, :] \leq \mathbf{u}[1] \frac{1}{d} + 0.5 \\ \vdots \\ \mathbf{1}^T \mathbf{H}_{XY}[n, :] \leq \mathbf{u}[n] \frac{1}{d} + 0.5 \end{array} \right\} \quad (20)$$

Here $\mathbf{1}^T \mathbf{H}_{XY}[i, :]$ denotes summation over the i^{th} row of the histogram \mathbf{H}_{XY} , i.e. $\mathbf{1}^T \mathbf{H}_{XY}[i, :] = \sum_{k=1}^m \mathbf{H}_{XY}[i, k]$, and $\mathbf{u}[i]$ is the i^{th} element of the array \mathbf{u} . Recall that the histogram \mathbf{H}_{XY} has n rows and m columns, where rows correspond to the bins and columns give the bin counts in unary encoding. The parameter d is a normalization factor that reads

$$d = \frac{\max\{\mathbf{u}[1], \dots, \mathbf{u}[n]\}}{m}. \tag{21}$$

Such a choice of the normalization parameter is to ensure that the r.h.s. of the inequalities (20) does not exceed the number of binary variables, m . The number 0.5 is to round the r.h.s. to a positive real number.

In this form the value of the normalization factor d can be obtained self-consistently during the optimization procedure. However, this would give a non-convex optimization problem due to the division in (21), which we wish to avoid. Therefore, we fix the normalization factor in our implementation to be a constant of our choice. We can check post-hoc, whether the solution has reached the bound on the number of unary digits in any of the bins of the histogram. If this is the case, the optimality has been affected and we can increase the bound and re-run the procedure. If this is not the case, the normalization does not affect the optimality.

Let us now discuss how the problem can be simplified by the tightening.

4.2.3 Problem tightening

The convergence of the BIP solver can be increased by introducing constraints that tighten the relaxation. For example, one can separate zeroes from ones in the histogram matrix \mathbf{H} with symmetry-breaking constraints:

$$H_{i,1} \geq H_{i,2} \geq H_{i,3} \geq \dots \geq H_{i,m-1} \geq H_{i,m}. \tag{22}$$

These constraints do not have any effect on the correctness of the solution but decrease size of the branch and bound tree of the solver.

Moreover, as we discussed in Sect. 2.2 (see expressions (6) and (7)), maximum and convolution describe the operation of a basic logic gate. Thus, it is natural to evaluate these operations simultaneously for each gate. Similar to (19), we can write for three binary variables, $x, y, z \in \{0, 1\}$:

$$\left. \begin{aligned} s &\leq x, \\ s &\leq y, \\ s &\leq z, \\ s &\geq x + y + z - 2. \end{aligned} \right\} \tag{23}$$

These constraints describe the max of two binary variables, x and y , and further convolution of the result with the third binary variable, z . Having discussed the atomic SSTA operations in unary encoding, we can proceed to the formulation of the SSTA Problem as a BIP problem, where we perform the same operation bit-wise on the unary-encoded counts.

4.3 Implementation and validation

Calculation of the circuit delay requires traversing the timing graph $G(E, V)$. Since a histogram \mathbf{H}_{sink} corresponding to the sink contains all binary variables obtained in the previous steps, it is natural to write the objective function of the BIP problem (4)–(5) as the sum of all the variables due to atomic operations in the graph, $\mathbf{1}^T \mathbf{H}_{\text{sink}} \mathbf{1}$, subject to constraints discussed above. Doing so, we obtain the SSTA problem of a Circuit *via* BIP as follows:

$$\begin{aligned}
 & \text{minimize} && \text{risk}(\mathbf{H}_{\text{sink}}) \\
 & \text{subject to} && \mathbf{H}_g[i, 1] \geq \dots \geq \mathbf{H}_g[i, m] \quad \forall g \in G : i = 1, \dots, n, \\
 & && \mathbf{1}^T \mathbf{H}_g[i, :] \leq \mathbf{u}[i] \frac{1}{d} + 0.5, \\
 & && s_g \leq \{x, y, z\}, \\
 & && s_g \geq x + y + z - 2, \\
 & && \mathbf{S}_g \leq \mathbf{G}_g
 \end{aligned} \tag{24}$$

where risk is a MILP representation of a risk measure. Notice that in many practical scenarios, one may consider the conditional value at risk (CVaR) (Rockafellar and Uryasev 2000) of the histogram–approximated random variable \mathbf{H}_{sink} as the risk measure $\text{risk}(\mathbf{H}_{\text{sink}})$, in an effort to “shift the probability mass left”, loosely speaking. In general, this would be implemented by summing up the counts in some number of “rightmost” bins of the histogram approximation. For testing purposes, we have utilized the expression $\mathbf{1}^T \mathbf{H}_{\text{sink}} \mathbf{1}$, which sums counts across the histogram, and implements CVaR at 0% confidence level.

This BIP formulation uses the symmetry–breaking constraints (22) and the 3–term multiplication model (23). Note that these constraints (23) imply taking max and convolution operations for a gate g , whereas constraints (19) correspond only to multiplication and, thus, should be added after each atomic operation, both max and convolution. Matrices \mathbf{G}_g in the BIP are unary representations of a delay of each gate, bounded from above elementwise by \mathbf{S}_g , again represented in unary.

To validate our BIP formulation of the SSTA, we have implemented (24) in Mosek 10.0 matrix-oriented API. As a test bench we have used a toy circuit used in Mishaghi and Blokhina (2020, Fig. 7) that gives a “ladder” sequence of logic gates. In terms of atomic operations, this sequence for the N th gate reads

$$\max\{ \underbrace{\dots \max[\max(\xi_1, \xi_2) + \xi_0, \xi_3] + \xi_0 \dots, \xi_{N+1}}_{N-1 \text{ times}} \} + \xi_0. \tag{25}$$

Here RVs ξ_i are drawn from the normal distribution, $\xi_i \sim \mathcal{N}(\mu_i, \sigma_i)$. Delay due to gates operation time (gate delay) is given by ξ_0 , and ξ_i ($i = 1, \dots, N + 1$) are inputs’ delays. Gate delays were assumed distributed according to the standard normal distribution, $\xi_0 \sim \mathcal{N}(1, 0)$; the mean values and standard deviations for inputs were drawn from the uniform distribution, following Ref. Mishaghi and Blokhina (2020).

This sequence was simulated (i) using Monte Carlo (MC) and (ii) by solving BIP (24). For each gate $g \in G(E, V)$, a new matrix \mathbf{H}_g containing binary variables was created as described above, until the sink was reached. Then, the final BIP problem

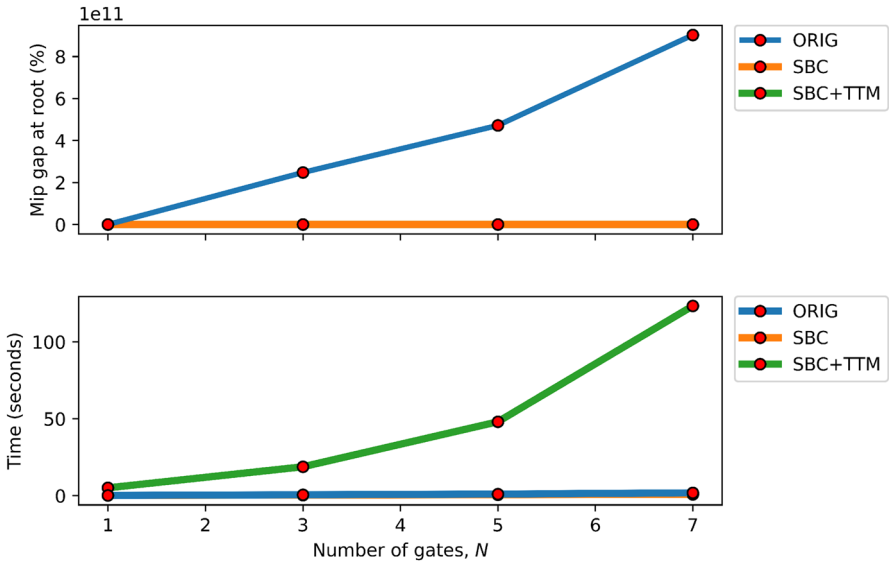


Fig. 2 Comparison of three BIP formulations for the SSTA. The methods are tested on a “ladder” of maxima with $n = 12$ bins, each bins’ count bounded from above by $m = 12$. The blue line indicates a method with only (19) constraints (ORIG), the orange line indicates a method with the symmetry-breaking constraints (SBC) (22) and the green line indicates a method with the symmetry-breaking constraints and a 3-term multiplication model (SBC+TTM) (23). In the first figure, the orange line overlaps the green line. In the second figure, the blue line overlaps the orange line

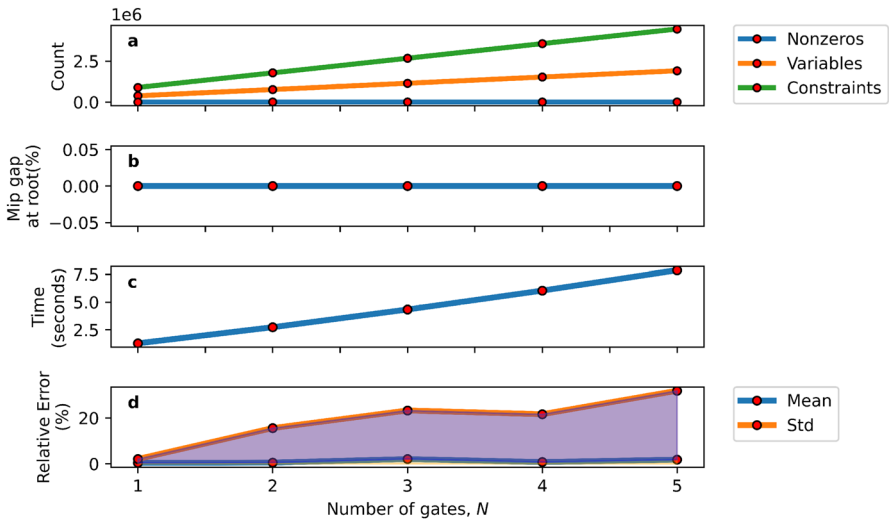


Fig. 3 Scalability of the model with the first relaxation (SBC) (22) tested on a “ladder” of maxima with $n = 25$ bins, $m = 20$ unary variables, and no time limit. The subplots show: **a**, the growth in the number of non-zeros (blue line), variables (orange line), and constraints (green line); **b**, MIP gap at a root node in percentage, MIP gap tolerance is set to 1%; **c**, time in seconds; **d**, relative error of the standard deviation (orange line) and mean (blue line) compared to Monte Carlo. (Color figure online)

was passed to a Mosek solver. Numerical experiments were ran on a machine equipped with Intel(R) Core(TM) i9-9880H (8 cores at 2.3 GHz and total of 16 threads) with 16 GB RAM.

The results are summarized in Figs. 2 and 3. Correctness of the BIP formulation can be seen from Fig. 3 where the mean and standard deviation of the sink delay distribution is compared against the MC. Although the number of constraints and variables scale linearly with the number of gates, these numbers are huge, thus, this approach does not scale. For example, the numbers of variables and constraints quickly reached the order of $\sim 10^6$ for only 5 gates (Fig. 3). In the next Section, we consider the GP formulation of SSTA, which scales much better.

5 SSTA via geometric programming

Next, we present a practical formulation of the SSTA problem. First, we will present the formulation by means of Geometric Programming, which is a restriction of the exact formulation, but where the error can be made arbitrarily small. Then, we will show its reformulation and scalability.

5.1 Statement of the problem

Naturally, we can treat the probability of each bin as a positive number in the range $[\epsilon, 1]$, where ϵ is a very small number. Multiplication of two bins leads to a monomial function of two variables and a neutral coefficient. Either the convolution or the maximum is then the sum of the multiplications, thus a posynomial. Therefore, general Algorithms 2 and 3 can be utilized within the Geometric Programming (GP) framework in a straightforward manner and no additional constraints are needed for the atomic operations.

In the following, we again consider a timing graph $G(E, V)$ consisting of N gates; the number of bins used for the histogram approximation of distributions is n . For each input gate g , we have a vector $\mathbf{e}_g \in \mathbb{R}_{++}^{n \times 1}$ created from generated numbers with Gaussian probability and a vector $\mathbf{z}_g \in \mathbb{R}_{++}^{n \times 1}$ of non-negative variables representing the bin probabilities. Similarly to Sect. 4, the geometric program starts with

$$\begin{aligned} & \text{minimize} && \text{risk}(\mathbf{z}_{\text{sink}}) \\ & \text{subject to} && \mathbf{e}_g \leq \mathbf{z}_g \leq \mathbf{1}, \quad g = 1, \dots, N, \end{aligned} \quad (26)$$

where $\text{risk}(\mathbf{z}_{\text{sink}})$ is a posynomially–representable risk measure of the delay \mathbf{z}_{sink} at the sink of the graph, and the bounds of the variables are standard GP-compatible inequalities. Subsequently, the construction of the geometric program follows the Algorithm 1 (“General SSTA algorithm”). In particular, for each maximum z_ζ of two histograms z_η and z_ξ , we constrain z as in Algorithm 2:

$$z_{\zeta}[i] = z_{\eta}[i] \cdot \sum_{k=1}^i z_{\xi}[k] + z_{\xi}[i] \cdot \sum_{k=1}^{i-1} z_{\eta}[k] \quad \forall i = 1 \dots n. \quad (27)$$

See Sect. 3.1 for a discussion. For each convolution z_{ζ} of two histograms z_{η} and z_{ξ} , we constrain elements of \mathbf{z} as in Algorithm 3:

$$z_{\zeta} = \sum_{i=1}^n \sum_{k=1}^{i-1} z_{\eta}[k] \cdot z_{\xi}[i - k] \quad (28)$$

up to the shifting. See Sect. 3.2 for further details.

5.2 Reformulation and relaxation

The posynomial formulation in (26) does not seem to be problematic in any way. Still, to improve its scalability, we may wish to consider a reformulation. In the following, we will concentrate only on convolution; the procedure is the same for the maximum.

After the first convolution in the last bin, we have a posynomial with $1 \cdot n$ terms (monomials with two variables). After the second convolution, we will have in the last bin a posynomial with $n \cdot n$ monomials each with three variables, and for the N^{th} gate we will have n^{N-1} monomials in the last bin, each with N variables. This clearly leads to an exponential growth in monomials after each convolution and maximum for a constant number of bins.

For each monomial in the posynomial, we need to introduce an exponential cone, two continuous variables, and two constraints. Thus, for a constant number of bins, the number of variables, cones, and constraints grows exponentially with the number of gates. For a constant number of gates $N + 1$, the number of variables, cones, and constraints grows as the sum of all bins $\sum_{i=1}^n i^N$ with the number of bins, which, in turn, can be expressed as a polynomial of degree $N + 1$ by Faulhaber's formula (Knuth 1993).

We can reduce this with the following simple trick: by introducing n new positive variables (monomials) and setting appropriate constraints. At the beginning of the traverse, we initiate two empty lists of vectors to store the successors and predecessors. After the convolution at the i^{th} gate, the resulting vector of posynomials \mathbf{z}_i is appended to the list of predecessors. A new vector of one-variable monomials is created and written in the successors' list, and constrained such that it is no less than the predecessor. This new vector represents a histogram and is propagated further. The last vector \mathbf{z}_{sink} that appears in the successors' list corresponds to the histogram of the delay at the sink node.

Subsequently, the formulation continues with the inequalities based on the equalities in Algorithms 2 and 3, while adding the auxiliary variables.³ In particular, for each convolution, we introduce $(n/2)(1+n)$ new exponential cones, thus $(2n/2)(1+n) + n$

³ For the implementation, see `maximum_GP_OPT` and `convolution_GP_OPT` in <https://github.com/bosakad/SSTA-via-GP/blob/experimental/src/timing/cvxdpyVariable.py#L273>.

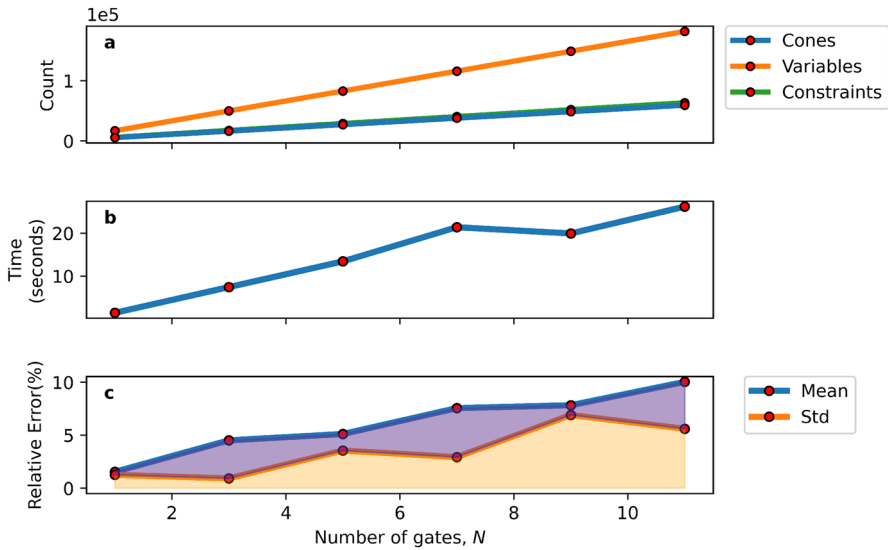


Fig. 4 Scalability of the GP (26) on a "ladder" of maxima with $n = 60$ bins and a varying number of gates, N . The subplots show: **a**, the growth in the number of cones (blue line), variables (orange line) and constraints (green line overlapped the blue line) as the number of gates, N , increases; **b**, time in seconds; **c**, relative error of the standard deviation (orange line) and mean (blue line) compared to Monte Carlo as the number of gates, N , increases. (Color figure online)

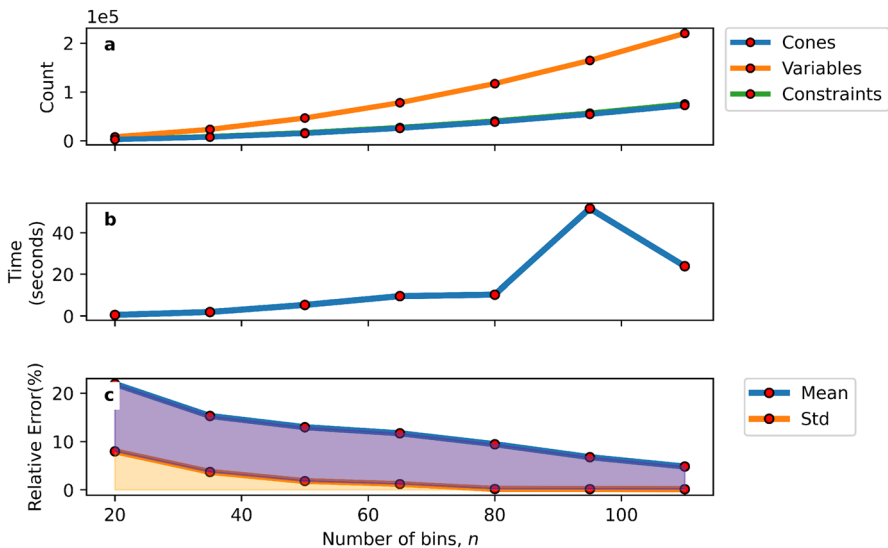


Fig. 5 Scalability of the reformulated GP (26) tested on a "ladder" of maxima with fixed $N = 4$ gates and varying numbers of bins, n , per gate. The subplots show: **a**, the growth in the number of cones (blue line), variables (orange line) and constraints (green line overlapping the blue line) as the number of bins, n , increases; **b**, time in seconds as the number of bins increases; **c**, relative error of the standard deviation (orange line) and mean (blue line) compared to Monte Carlo, as the number of bins, n , increases. (Color figure online)

auxiliary variables. Such a reformulation gives the exact same solution as the original GP. We just decreased the exponential growth of variables, cones and constraints to a linear one with the number of gates, and a high-degree polynomial growth with the number of bins to always quadratic. The values are slightly different for the maximum, but the asymptotics before and after the reformulation are identical.

Notice that (26) can be transformed into a standard GP-compatible inequality. Thus, the reformulation of (26) remains a generalized GP program.

5.3 Implementation and validation

We have prototyped the formulations in CVXPY Agrawal et al. (2018); Diamond and Boyd (2016). For benchmarking purposes, we have passed the instances to MOSEK 10.0, which ran on a laptop equipped with Intel(R) Core(TM) i9-9880H (8 cores at 2.3 GHz and total of 16 threads) with 16 GB RAM. The same toy circuit was used as in Sect. 4.3.

The scalability of the reformulated GP model is demonstrated in Figs. 4 and 5. The results on a ladder of maxima parameterized by the depth of the ladder are shown in Fig. 4. Notice that the numbers of cones (blue line) and variables (orange line) are linear in the depth of the ladder. At the same time, the relative error increases.

Figure 5 demonstrates the scalability of the reformulated GP (26) on a ladder of maxima and convolutions, parameterized by the number of bins per gate. Notice that the numbers of cones (blue line) and variables (orange line) are quadratic in the number of bins. At the same time, the relative error decreases with the number of bins, as expected.

6 Discussion and conclusions

In this paper, the problem of the calculation of the maximum delay in a digital circuit under uncertainty (also known as SSTA) is studied from the mathematical optimization point of view *for the first time*. Using a histogram representation of the delays distributions for simplicity, we have presented two formulations of SSTA as an optimization problem. Section 4 shows Binary Integer Programming (BIP) approach, which is a formal formulation and does not scale. Section 5 gives a more practical formulation of SSTA as a Geometric Programming (GP) problem.

For a reformulation of the GP, we have demonstrated linear scaling with the number of gates and quadratic scaling with the number of bins. The SSTA has been successfully computed using 30 bins for a circuit with 400 gates in 440 s which ran on an 8-processor machine equipped with Intel Xeon Scalable Platinum 8160 (192 cores at 2.1GHz and 384 hardware threads) with 1536 GB RAM.⁴

The histogram approximation, which has been previously studied (Liou et al. 2001) as a replacement of Monte Carlo simulations, is used in this work for optimization purposes. However, this approach has clear disadvantages: (i) as we increase the number of gates, we have to increase the size of the interval, and with that the number of

⁴ Full code used in this research can be found in the repository <https://github.com/bosakad/GP-Optimization/>.

Table 1 An overview of the approaches that approximate probability distributions with various parametric classes of distributions, with references to their uses in replacements of Monte Carlo in simulation, and our suggestions as to the classes of optimization problems obtained using the technique of Sect. 4

Model	Simulation refs	Optimization formulation
Histogram	Liou et al. (2001)	Geometric program of Sect. 5
Impulse train	Naidu (2002)	Mixed-integer linear
Gaussian comb	Mishagli and Blokhina (2020)	Mixed-integer tame
Gaussian mixtures	Mishagli and Blokhina (2020)	Mixed-integer tame

bins; (ii) we also assume the support of the delay distribution is known⁵ (iii) Also, it should be noted that the correlations between the delays were not taken into account.

On the other hand, this approach allowed us to (i) perform the robust optimization of delays' distributions, unlike in other statistical approaches, where only the statistical moments are taken into account, and (ii) perform computations in polynomial time up to any fixed precision using GP. Last but not least, the histogram formulation of the SSTA makes the results transparent and easy to understand.

Some of these challenges could be addressed. Similar to the approximation algorithm (Liou et al. 2001, Section 3.3) of Liou et al., one could address the scalability issue at the cost of some error by decomposing the circuit into “supergates”, possibly hierarchically, and at the further cost of estimating only the tail of the delay distribution by (i) “filtering out unnecessary stems”, i.e., discarding sample paths, which are guaranteed not to influence the tail of the distribution. See Liou et al. (2001, Section 2.3) for suggestions how this could be performed. The same approach of Liou et al. (2001, Section 2.3) could also be used to address the challenge (ii) above, as it can be used to estimate the range of arrival times of events. The challenge (iii) above, phrased in terms of addressing the correlations seems to be inherently difficult, albeit perhaps less important. This inherent difficulty extends to measuring the correlations between more than two gates' delays, especially when conditioned on external factors such as a change of temperature. Indeed, many sources (Jyu et al. 1993, e.g., p. 131) claim “cell library designers agree that it is reasonable to expect the delays for components on a single chip to track each other.”

Let us now chart some potential avenues for further research. Easily, one could replace the uniform distribution centered at the midpoint of each bin with a triangular distribution centered at the midpoint of each bin. In his pioneering paper (Naidu 2002), Naidu used such “impulse train” distributions as a replacement of Monte Carlo for simulation purposes. Our binary–integer optimization formulation of Sect. 4 should be easy to extend to the triangular distributions, and would remain mixed-integer linear. Whether the geometric–programming approach of Sect. 5 would be as easy to extend, remains to be investigated. Either way, this would be an interesting extension.

⁵ The delay in a circuit is clearly bounded from below and may well be bounded from above by considerations of practicality with respect to clocking frequency. Principled methods for establishing upper bounds have been proposed [Section 2.3 Liou et al. (2001)], but we have used trial and error to set the upper bound in the proof-of-concept implementation in this paper.

Further, our work could be plausibly extended to the Gaussian comb model of Mishaghi and Blokhina (2020), where one would replace the uniform distribution centered at the midpoint of each bin with a Gaussian kernel function. As such, the Gaussian comb model is a very special case of the Gaussian mixture model with predefined, uniformly distributed expectations of the components. There, one optimizes over the mixture coefficients, rather than the counts in a histogram, and the objective function has a more complicated form (which expresses the integral in a closed-form). We conjecture the resulting mixed-integer non-linear optimization problems are “tame” in their continuous part, i.e., the continuous part is definable in an o-minimal structure. This is a fast-growing area of optimization, due to the applications in deep learning (see, e.g., Bolte and Pauwels (2021)), but mixed-integer extensions do not seem to have been studied yet, and there certainly are no off-the-shelf solvers.

More broadly, one could consider Gaussian mixture models or infinitely-smooth radial basis functions (RBFs). Infinitely smooth RBF, such as Multiquadric RBF or Inverse quadratic RBF, are real-analytic ($C^\infty(\mathbb{R})$), and hence one can formulate mixed-integer analytical optimization problems over these. While optimization over real-analytic functions is becoming better understood (Absil and Kurdyka 2006; Kurdyka et al. 2000), these optimization problems seem very challenging. While the famous result of Kurdyka et al. (2000) shows the finite length of the gradient flows, local minima are not necessarily stable equilibria of the gradient-descent system, and vice versa [Proposition 2 Absil and Kurdyka (2006)]. That is: local minimality is neither necessary nor sufficient for stability. Integer analytic optimization hence seems difficult to work with, other than using spatial branch-and-bound (Smith and Pantelides 1999), which may not be sufficiently scalable.

Future work may also involve an extension of the maximum computation and the gate sizing program for the case of correlated random variables and related problems of circuit design.

Funding Open Access funding provided by the IReL Consortium The research that led to these results received funding from OP RDE under Grant Agreement No CZ.02.1.01/0.0/0.0/16_019/0000765.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

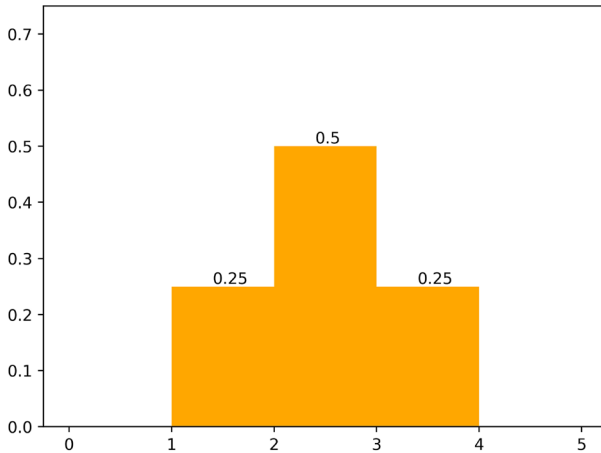


Fig. 6 An example histogram with $n = 5$ bins; width of the bin is 1

Appendix: Note on unary encoding

The idea behind the unary encoding is as follows. A histogram, which is an array of real numbers, can be written as a matrix of *binary numbers*. Each bin is then expressed by a row in the matrix. In this case, the probability is given by a sum of the row elements divided by both a sum of the matrix elements and a width of the bin (for normalization purposes).

Consider a toy example shown in Fig. 6. This histogram can be represented by a 5×3 matrix as follows:

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Note that not all real numbers can be encoded by a finite number of binary numbers. The accuracy of the encoding is proportional to the number of columns in the binary representation.

Numerical example. Consider two histograms:

$$\mathbf{H}_A = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{H}_B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (29)$$

The convolution \mathbf{H}_{AB} of these histograms, according to the Algorithm 4, will require introducing the constraints (19) for the unary numbers in \mathbf{H}_A and \mathbf{H}_B . For example,

for $z = 3$ we have $k = 1, 2$, and there will be the following values

$$\begin{aligned}\mathbf{H}_A[k, i] &= \{\mathbf{H}_A[1, i], \mathbf{H}_A[2, i]\}, & i &= 1, 2, 3 \\ \mathbf{H}_B[z - k, i] &= \{\mathbf{H}_B[2, j], \mathbf{H}_B[1, j]\}, & j &= 1, 2, 3\end{aligned}$$

The constraints (19) are needed to be introduced for the permutation of these sets. Then, the constraints are passed to a solver together with the auxiliary variables s .

References

- Absil PA, Kurdyka K (2006) On the stable equilibrium points of gradient systems. *Syst Control Lett* 55(7):573–577. <https://doi.org/10.1016/j.sysconle.2006.01.002>
- Agrawal A, Verschueren R, Diamond S, Boyd S (2018) A rewriting system for convex optimization problems. *J Control Decis* 5(1):42–60. <https://doi.org/10.1080/23307706.2017.1397554>
- Beece DK, Visweswariah C, Xiong J, Zolotov V (2014) Transistor sizing of custom high-performance digital circuits with parametric yield considerations. *Optim Eng* 15(1):217–241. <https://doi.org/10.1007/s11081-012-9208-0>
- Berkelaar MRCM, Jess JAG (1990) Gate sizing in MOS digital circuits with linear programming. In: *Proceedings of the conference on European design automation, EURO-DAC '90*, pp 217–221. IEEE Computer Society Press, Washington, DC, USA
- Bhasker J, Chadha R (2009) *Static timing analysis for nanometer designs. A practical approach*. Springer, Berlin
- Blaauw D, Chopra K, Srivastava A, Scheffer L (2008) Statistical timing analysis: from basic principles to state of the art. *IEEE Trans Comput Aided Des Integr Circuits Syst*. <https://doi.org/10.1109/TCAD.2007.907047>
- Bolte J, Pauwels E (2021) Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Math Program* 188(1):19–51. <https://doi.org/10.1007/s10107-020-01501-5>
- Boyd SP, Kim SJ, Patil DD, Horowitz MA (2005) Digital circuit optimization via geometric programming. *Oper Res* 53(6):899–932. <https://doi.org/10.1287/opre.1050.0254>
- Boyd S, Kim SJ, Vandenberghe L, Hassibi A (2007) A tutorial on geometric programming. *Optim Eng* 8(1):67–127. <https://doi.org/10.1007/s11081-007-9001-7>
- Brenner U, Vygen J (2009) Analytical methods in placement. In: *Handbook of algorithms for physical design automation*. Auerbach Publications. Num Pages: 20
- Chang H, Sapatnekar S (2005) Statistical timing analysis under spatial correlations. *IEEE Trans Comput Aided Des Integr Circuits Syst* 24(9):1467–1482. <https://doi.org/10.1109/TCAD.2005.850834>
- Chang H, Sapatnekar SS (2003) Statistical timing analysis considering spatial correlations using a single PERT-like traversal. In: *Proceedings of the ICCAD*, pp 621–625. <https://doi.org/10.1109/ICCAD.2003.159746>
- Chang H, Zolotov V, Narayan S, Visweswariah C (2005) Parameterized block-based statistical timing analysis with non-gaussian parameters, nonlinear delay functions. In: *Proceedings of the 42nd annual design automation conference, DAC '05*, pp 71–76. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/1065579.1065604>
- Cheng L, Gong F, Xu W, Xiong J, He L, Sarrafzadeh M (2012) Fourier series approximation for max operation in non-gaussian and quadratic statistical static timing analysis. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 20(8):1383–1391. <https://doi.org/10.1109/TVLSI.2011.2157843>
- Cheng L, Xiong J, He L (2007) Non-linear statistical static timing analysis for non-gaussian variation sources. In: *2007 44th ACM/IEEE design automation conference*, pp 250–255. ISSN: 0738-100X
- Clark CE (1961) The greatest of a finite set of random variables. *Oper Res* 9(2):145–162. <https://doi.org/10.1287/opre.9.2.145>. Place: Institute for Operations Research and the Management Sciences (INFORMS), Linthicum, Maryland, USA Publisher: INFORMS
- Diamond S, Boyd S (2016) CVXPY: a python-embedded modeling language for convex optimization. *J Mach Learn Res* 17(1):2909–2913

- Forzan C, Pandini D (2009) Statistical static timing analysis: a survey. *Integr VLSI J* 42(3):409–435. <https://doi.org/10.1016/j.vlsi.2008.10.002>
- Freeley J, Mishaghi D, Brazil T, Blokhina E (2018) Statistical simulations of delay propagation in large scale circuits using graph traversal and kernel function decomposition. In: Proceedings of the SMACD
- Held S, Korte B, Rautenbach D, Vygen J (2011) Combinatorial optimization in VLSI design. *Comb Optim*. <https://doi.org/10.3233/978-1-60750-718-5-33>
- Jacobs ETAF, Berkelaar MRCM (2000) Gate sizing using a statistical delay model. In: Proceedings design, automation and test in Europe conference and exhibition 2000, pp 283–290. <https://doi.org/10.1109/DATE.2000.840285>
- Jess JAG, Kalafala K, Naidu SR, Otten RHJM, Visweswariah C (2003) Statistical timing for parametric yield prediction of digital integrated circuits. In: Proceedings of the DAC. ACM, pp 932–937. <https://doi.org/10.1145/775832.776066>. Event-place: Anaheim, CA, USA
- Jess JAG, Kalafala K, Naidu SR, Otten RHJM, Visweswariah C (2006) Statistical timing for parametric yield prediction of digital integrated circuits. *IEEE Trans Comput Aided Des Integr Circuits Syst* 25(11):2376–2392. <https://doi.org/10.1109/TCAD.2006.881332>
- Joshi S, Boyd S (2008) An efficient method for large-scale gate sizing. *IEEE Trans Circuits Syst I Regular Pap* 55(9):2760–2773. <https://doi.org/10.1109/TCSI.2008.920087>
- Jyu HF, Malik S, Devadas S, Keutzer K (1993) Statistical timing analysis of combinational logic circuits. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 1(2):126–137. <https://doi.org/10.1109/92.238423>
- Khandelwal V, Srivastava A (2005) A general framework for accurate statistical timing analysis considering correlations. In: Proceedings of the DAC, pp 89–94. <https://doi.org/10.1145/1065579.1065607>. ISSN: 0738-100X
- Knuth DE (1993) Johann Faulhaber and sums of powers. *Math Comput* 61(203):277–294. <https://doi.org/10.1090/S0025-5718-1993-1197512-7>
- Korte B, Vygen J (2008) Combinatorial problems in chip design. In: Grötschel M, Katona GOH, Sági G (eds) Building bridges: between mathematics and computer science, Bolyai society mathematical studies. Springer, Berlin, pp 333–368. https://doi.org/10.1007/978-3-540-85221-6_12
- Kurdyka K, Mostowski T, Parusinski A (2000) Proof of the gradient conjecture of R. Thom. *Ann Math* 152(3):763–792. <https://doi.org/10.2307/2661354>
- Liou JJ, Cheng KT, Kundu S, Krstic A (2001) Fast statistical timing analysis by probabilistic event propagation. In: Proceedings of the 38th design automation conference (IEEE Cat. No.01CH37232), pp 661–666. <https://doi.org/10.1145/378239.379043>
- Mishaghi D, Blokhina E (2020) Radial basis functions based algorithms for non-gaussian delay propagation in very large circuits. In: Krzhizhanovskaya VV, Závodszy G, Lees MH, Dongarra JJ, Sloat PMA, Brissos S, Teixeira J (eds) Computational science—ICCS 2020. Lecture notes in computer science. Springer, Cham, pp 217–229. https://doi.org/10.1007/978-3-030-50426-7_17
- Naidu S (2002) Timing yield calculation using an impulse-train approach. In: Proceedings of ASP-DAC/VLSI design 2002. 7th Asia and South Pacific design automation conference and 15th international conference on VLSI design, pp 219–224. <https://doi.org/10.1109/ASPDAC.2002.994923>
- Naidu SR (2021) A convex programming solution for gate-sizing with pipelining constraints. *Optim Eng*. <https://doi.org/10.1007/s11081-021-09616-0>
- Rakai L, Farshidi A (2015) Sizing digital circuits using convex optimization techniques. In: Fakhfakh M, Tlelo-Cuautle E, Siarry P (eds) Computational intelligence in digital and network designs and applications. Springer, Cham, pp 3–32. https://doi.org/10.1007/978-3-319-20071-2_1
- Ramprasath S, Vijaykumar M, Vasudevan V (2016) A skew-normal canonical model for statistical static timing analysis. *IEEE Trans Very Large Scale Integr (VLSI) Syst* 24(6):2359–2368. <https://doi.org/10.1109/TVLSI.2015.2501370>
- Rockafellar RT, Uryasev S et al (2000) Optimization of conditional value-at-risk. *J Risk* 2:21–42
- Sapatnekar S (2004) *Timing*. Springer, Berlin
- Sapatnekar S, Rao V, Vaidya P, Kang SM (1993) An exact solution to the transistor sizing problem for CMOS circuits using convex optimization. *IEEE Trans Comput Aided Des Integr Circuits Syst* 12(11):1621–1634. <https://doi.org/10.1109/43.248073>
- Singh J, Sapatnekar SS (2008) A scalable statistical static timing analyzer incorporating correlated non-gaussian and gaussian parameter variations. *IEEE Trans Comput Aided Des Integr Circuits Syst* 27(1):160–173. <https://doi.org/10.1109/TCAD.2007.907241>

- Smith EMB, Pantelides CC (1999) A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Comput Chem Eng* 23(4):457–478. [https://doi.org/10.1016/S0098-1354\(98\)00286-5](https://doi.org/10.1016/S0098-1354(98)00286-5)
- Vielma JP, Nemhauser GL (2011) Modeling disjunctive constraints with a logarithmic number of binary variables and constraints. *Math Program* 128:49–72
- Visweswariah C (2003) Death, taxes and failing chips. In: *Proceedings of the DAC*. IEEE, pp 343–347. <https://doi.org/10.1145/775832.775921>. Event-place: Anaheim, CA, USA
- Visweswariah C, Ravindran K, Kalafala K, Walker SG, Narayan S, Beece DK, Piaget J, Venkateswaran N, Hemmett JG (2006) First-order incremental block-based statistical timing analysis. *IEEE Trans Comput Aided Des Integr Circuits Syst* 25(10):2170–2180. <https://doi.org/10.1109/TCAD.2005.862751>
- Visweswariah C, Ravindran K, Kalafala K, Walker SG, Narayan S (2004) First-order incremental block-based statistical timing analysis. In: *Proceedings of the DAC*. ACM, New York, NY, USA, pp 331–336. <https://doi.org/10.1145/996566.996663>. Event-place: San Diego, CA, USA
- Vygen J (2006) Slack in static timing analysis. *IEEE Trans Comput Aided Des Integr Circuits Syst* 25(9):1876–1885. <https://doi.org/10.1109/TCAD.2005.858348>
- Wolsey L, Nemhauser G (1999) *Integer and combinatorial optimization*. Wiley series in discrete mathematics and optimization. Wiley, New York
- Zhang L, Chen W, Hu Y, Gubner JA, Chen CCP (2005) Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model. In: *Proceedings of the DAC*, pp 83–88. <https://doi.org/10.1109/DAC.2005.193778>. ISSN: 0738-100X
- Zhang L, Hu Y, Chen CCP (2006) Statistical timing analysis with path reconvergence and spatial correlations. In: *Proceedings of the DAC*, vol 1, p 5. <https://doi.org/10.1109/DATE.2006.243890>. ISSN: 1530-1591
- Zhan Y, Strojwas AJ, Li X, Pileggi LT, Newmark D, Sharma M (2005) Correlation-aware statistical timing analysis with non-Gaussian delay distributions. In: *Proceedings of the DAC*, pp 77–82. <https://doi.org/10.1145/1065579.1065605>. ISSN: 0738-100X

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.