Correlation-Aware Heuristics for Evaluating the Distribution of the Longest Path Length of a DAG with Random Weights

Louis-Claude Canon, Emmanuel Jeannot

Correlation-Aware Heuristics for Evaluating the Distribution of the Longest Path Length of a DAG with Random Weights

Louis-Claude Canon
FEMTO-ST / CNRS and the Université de Franche-Comté, Besançon, France
louis-claude.canon@univ-fcomte.fr

Emmanuel Jeannot
LaBRI and Inria Bordeaux Sud-Ouest
Talence, France
emmanuel.jeannot@inria.fr

December 16, 2016

Abstract

Coping with uncertainties when scheduling task graphs on parallel machines requires to perform non-trivial evaluations. When considering that each computation and communication duration is a random variable, evaluating the distribution of the critical path length of such graphs involves computing maximums and sums of possibly dependent random variables. The discrete version of this evaluation problem is known to be #P-hard. Here, we propose two heuristics, CorLCA and Cordyn, to compute such lengths. They approximate the input random variables and the intermediate ones as normal random variables, and they precisely take into account correlations with two distinct mechanisms: through lowest common ancestor queries for CorLCA and with a dynamic programming approach for Cordyn. Moreover, we empirically compare some classical methods from the literature and confront them to our solutions. Simulations on a large set of cases indicate that CorLCA and Cordyn constitute each a new relevant trade-off in terms of rapidity and precision.

Keywords: stochastic scheduling, graph heuristic, PERT

1 Introduction

Evaluating the execution time (makespan) of a parallel application modeled by a task graph is an important problem in scheduling theory. This problem is simple to solve in a deterministic setting. However, modern parallel systems are not fully deterministic and may be subject to many kinds of uncertainties: executions may fail; outcomes can be corrupted (e.g., network error); and, tasks or communications durations vary because of imprecise predictions (due to system noise, network congestion or input sensitiveness).

This paper focuses on duration uncertainties as it concerns the main inputs of a scheduling problem. Durations are modeled with random variables instead of deterministic values to ensure a precise description of the overall system. Evaluating the performance of a static scheduling procedure–by performing maximums and sums over durations–becomes a difficult operation because of the dependencies between random variables arising from the graph structure. In particular, no simple method exists for evaluating the distribution of the maximum of dependent random variables. The discrete version of this problem (when all input random variables are discrete with only rational values) was proved to be #P-hard[1] [17].

In this paper, we propose two new heuristics to address this evaluation problem. Many methods (either exact or approximate) exist but none pro-

[1] #P is the class of counting problems that correspond to NP decision problems.

vides the rapidity/precision trade-off required for heavy use in another procedure such as a static scheduling algorithm that needs to evaluate numerous partial solutions. Indeed, this work is motivated by the need to design a building block that efficiently computes the makespan distribution in the context of stochastic scheduling. In [9], we have designed Rob-HEFT, a heuristic which assigns each task greedily to the best processor by testing each possible allocation (the task is assigned to the machine for which its completion time is the best). Thus, Rob-HEFT requires to evaluate many partial solutions precisely and rapidly. But, other heuristics from the literature could benefit from the proposed solutions such as SHEFT [40], MCS [49] and SDLS [24].

These proposed methods are evaluated against classical techniques of the literature that compute operations (sums and maximums) on random variables when there are dependencies. These techniques are gathered from the scheduling literature but also from other areas of computer science that consider variations of this problem (project management [26] and digital circuit design [5]). Actually, although this paper is focused on scheduling and parallelism, the proposed results are applicable in these other fields.

This article is organized as follows. In Section 2, we formalize the general problem and show its relation to the scheduling problematic. Many contributions have been proposed for this problem in the literature and we present some significant approaches in Section 3. We propose the new methods in Section 4 and they are empirically evaluated in Section 5.

2 Model and Problem Definition

This section defines the problem and we show that some of its variations are equivalent. We first define the type of random variables that we use and show the relation with a specific scheduling problem that can be reduced to the general problem. Notations are summarized in Table 1.

2.1 Random Variable

Let η be a random variable. Its probability density function is f_η and is defined on $\mathbb{R}$. Its cumulative distribution function is $F_\eta(x) = \int_{-\infty}^{x} f_\eta(x)dx$. This function F_η gives the probability that η takes a value lower than or equal to a given constant, i.e., $F_\eta(x) = \Pr[\eta \leq x]$. Finally, the expected value of η is noted μ_η and its standard deviation is noted σ_η.

2.2 Longest Path of a Directed Acyclic Graph with Random Weights

Let $G = (V, E, \mathcal{X})$ be a DAG (Directed Acyclic Graph). Each vertex and each edge is weighted by a random variable. The weight of vertex $v_i \in V$ is noted $X_i \in \mathcal{X}$ and the weight of edge $(v_i, v_j) \in E$ is $X_{ij} \in \mathcal{X}$. Graph G contains n vertices and m edges (i.e., $|V| = n$, $|E| = m$ and $|\mathcal{X}| = m + n$). The vertices v_1, v_2, $\ldots$, v_n are ordered in a topological order[2]. Without loss of generality, we assume there are a single source and a single sink. The graph models a parallel application where vertices are tasks and edges are communications or synchronizations between tasks.

The graph structure encodes an arithmetic expression on the weights. To compute this expression we define two types of *intermediate results*: Y_j, which is the intermediate results for vertex v_j, and $Y_{i,j}$, which is the intermediate results for edge (v_i, v_j). Formally, for each vertex $v_j \in V$, we define the random variable $Y_j = X_j + \max_{v_i \in \mathrm{Pred}(v_j)} Y_{ij}$ (i.e., a maximum is performed when several edges target the same vertex). Similarly, for each edge $(v_i, v_j) \in E$, $Y_{ij} = X_{ij} + Y_i$ (i.e., the weights that are present on a given path are added). Both operations are represented on Figure 1. The intermediate result

[2] In a topological order, vertex indexes are ordered such that $\forall (v_i, v_j) \in V^2, i < j \Rightarrow (v_j, v_i) \notin E$.

Table 1: Notation summary.

Symbol	Definition
$G = (V, E, \mathcal{X})$	directed acyclic graph
$V = \{v_i : i \in [1..n]\}$	set of vertices
E	set of edges
n	number of vertices ($n = \|V\|$)
m	number of edges ($m = \|E\|$)
$\mathrm{Pred}(v_i)$	set of predecessors of vertex $v_i \in V$ ($\mathrm{Pred}(v_i) \subset V$)
$\mathrm{Succ}(v_i)$	set of successors of vertex $v_i \in V$ ($\mathrm{Succ}(v_i) \subset V$)
$\mathcal{X}$	set of random variables ($\|\mathcal{X}\| = n + m$)
X_i	weight of vertex $v_i \in V$ ($X_i \in \mathcal{X}$)
X_{ij}	weight of edge $(v_i, v_j) \in E$ ($X_{ij} \in \mathcal{X}$)
Y_i	intermediate result of vertex $v_i \in V$ $\left(Y_i = X_i + \max_{v_j \in \mathrm{Pred}(v_i)} Y_{ji}\right)$
Y_{ij}	intermediate result of edge $(v_i, v_j) \in E$ ($Y_{ij} = X_{ij} + Y_i$)
Y_n	final result
f_η	probability density function of random variable η
F_η	cumulative distribution function of random variable η ($F_\eta(x) = \Pr[\eta \leq x]$)
μ_η	expected value of random variable η
σ_η	standard deviation of random variable η
$\eta \sim \mathcal{N}(\mu_\eta, \sigma_\eta)$	a random variable η follows a normal law with expected value μ_η and standard deviation σ_η
$\rho_{\eta,\varepsilon}$	correlation coefficient between random variables η and ε

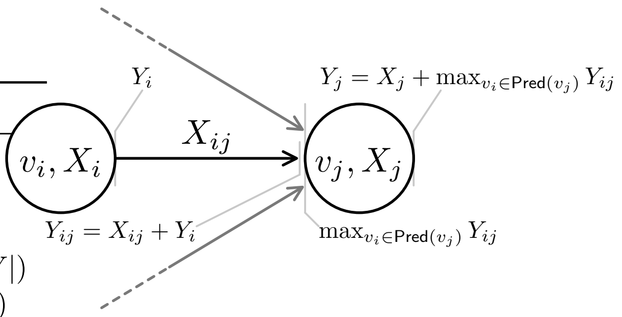

Y_i
$Y_j = X_j + \max_{v_i \in \mathsf{Pred}(v_j)} Y_{ij}$
X_{ij}
v_i, X_i
v_j, X_j
$Y_{ij} = X_{ij} + Y_i$
$\max_{v_i \in \mathsf{Pred}(v_j)} Y_{ij}$

Figure 1: Intermediate result in a sub-graph of two vertices (v_i and v_j). The arithmetic operations that are performed are: a sum to compute Y_{ij} (at the end of edge (v_i, v_j)), the maximum for Y_j (maximum of all incoming edges to the vertex) and a sum with the weight of vertex X_j.

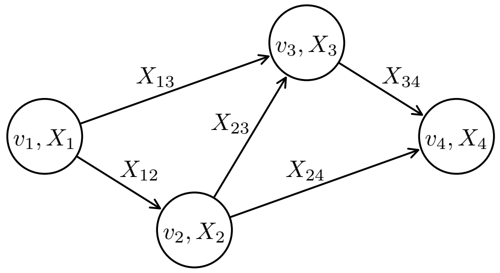


Figure 2: Graph with four vertices (v_1, v_2, v_3 and v_4) and with vertex and edge weights.

of the sink Y_n is the final result of the arithmetic expression denoted by the graph.

2.3 Problem Definition

The problem we consider consists in determining the probability law of this last random variable Y_n. More precisely, we want to determine the probability that the longest path length takes a value lower than or equal to some given constant.

Figure 2 illustrates some arithmetic operations represented in a graph. The intermediate result that corresponds to vertex v_2 is $Y_2 = X_2 + Y_{12}$. As $Y_{12} = X_{12} + Y_1$ and $Y_1 = X_1$, then $Y_2 = X_2 + X_{12} + X_1$. A maximum is performed when evaluating the intermediate result of vertex v_3. Hence, $Y_3 = X_3 + \max(Y_{23}, Y_{13})$. We can express Y_3 using only weights in $\mathcal{X}$. We obtain $Y_3 = X_3 + \max(X_{23} + X_2 + X_{12} + X_1, X_{13} + X_1)$. The encoded expression in this graph is given by the final result Y_4 (i.e., the intermediate result of the sink, v_4). It is $Y_4 = X_4 + \max(X_{24} + X_2 + X_{12}, X_{34} + X_3 + \max(X_{23} + X_2 + X_{12}, X_{13})) + X_1$. We have factorized X_1 in the maximums and the formula cannot be factorized any more. The problem consists in characterizing the distribution of Y_4 given the random variables in set $\mathcal{X}$.

Note that some arithmetic expressions in a max+ algebra cannot be represented by a graph. For instance, it is not possible to encode the expression $X + X$, where X is a given random variable. The considered arithmetic expressions have thus a specific structure. For instance, random variables that are added are all independent.

2.4 Scheduling with Random Durations and Bounded Resources

We present here an application of this problem in the context of task scheduling. We consider the general case when tasks are subject to precedence constraints with bounded resources. Task durations are specified by random variables. The total duration of a static schedule is thus also a random variable and its evaluation can be reduced to the problem studied in this paper.

This reduction is obtained by remarking that a task allocated to a specific processor cannot start its execution until all its predecessors have terminated theirs and the considered processor has finished its previous tasks. We have thus two kinds of precedences: those that come from the task graph and those that are related to the order in which tasks are executed on each processor. The constraints of the second kind correspond to additional edges in the task graph (the final graph remains acyclic) and are related to the anteriority enforced by the schedule. Therefore, we are able to deal with schedules for bounded resources: once the schedule is computed, we enforce sequentiality on the resources by adding such edges of zero weight between tasks on the same resources (see Figure 3). The maximum of the end dates of all the tasks of this graph is the makespan of the schedule.

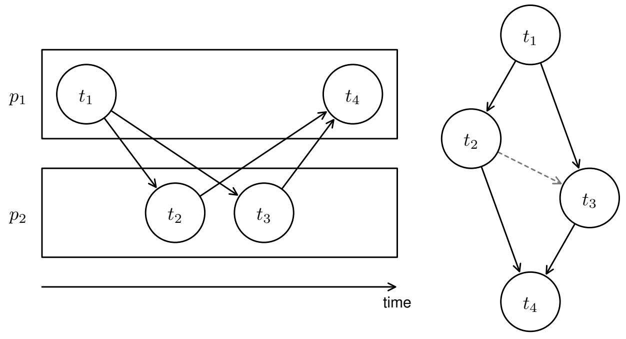


Figure 3: Four tasks (t_1, t_2, t_3 and t_4) are scheduled on two processors (p_1 and p_2). The corresponding graph contains one additional edge (in dash line) that is not present in the input task graph.

The start date of a task execution is obtained by performing a maximum on a set of execution end dates. Then, an end date is the result of a sum over a random duration and a start date. The problem consists finally in evaluating the distribution of the end date of the last finished task (the makespan of the schedule). Characterizing the distribution of the completion time of a set of dependent jobs with random durations is therefore equivalent to evaluating the distribution of the length of the longest path of a DAG with random weights.

An example of four tasks scheduled on two processors is proposed on Figure 3. Task t_1 has no predecessor and is thus the first to start its execution. Tasks t_2 and t_3 both depend on task t_1 and cannot start their executions before t_1 finishes its own. Finally, task t_4 depends on the two previous tasks. Tasks t_1 and t_4 are executed on processor p_1 and tasks t_2 and t_3 are executed on p_2. As the execution of t_2 is anterior to the execution of t_3, an edge is added between these two tasks in the corresponding graph. In this example, task durations (i.e., weights) are not represented. The duration of each task is associated to the corresponding vertex in the graph.

2.5 Dependency between Intermediate Results

In order to determine the intermediate result of a vertex $v_j \in V$, the expression $Y_j = X_j + \max_{v_i \in \mathrm{Pred}(v_j)} Y_{ij}$ must be evaluated. Operands of any maximum are always intermediate results. The main difficulty revealed by related works concerns the dependency between all the intermediate results. If they were independent, evaluating the distribution of the longest path length would indeed be easy using the methods presented later in Section 3.1.1.

Figure 2 illustrates this phenomenon. The intermediate result of vertex v_3 is formulated $X_3 + \max(Y_{23}, Y_{13})$. The operands of this maximum, Y_{23} and Y_{13}, are dependent because they both are expressed using the same random variable, X_1.

In the literature, the problem raised by operand dependencies is also called *path reconvergence* [48], *shared activity jias* [46] or *topological dependency* [5].

2.6 Representation Equivalence

In some contexts, there is either no weight on the vertices or no weight on the edges. This is the case when managing projects represented with an activity on arc (AoA) network or with an activity on node (AoN) network. Transforming an AoN network into an AoA network while minimizing the number of additional arcs is NP-Hard [23]. However, this transformation may be done in polynomial time and space when the minimization is not required. In particular, we can transform an instance of our problem into an instance with no weight on the vertices in polynomial time [29]. Each vertex must be replaced by a pair of vertices connected by an edge whose weight is the same as the weight of the initial vertex. The first (resp., second) vertex of this pair becomes a successor (resp., predecessor) of all the predecessors (resp., successors) of the initial vertex. The number of vertices is increased by a factor of two through this transformation. Moreover, the transformation complexity is linear in the number of edges m. An analogous linear algorithm exists to convert an instance of our problem into an instance with no weight on the edges.

These representations are thus equivalent and we specify whenever it is necessary if there is no weight on the vertices or on the edges.

3 State of the Art and Related Work

We begin by presenting different mechanisms for evaluating the result of an arithmetic operation on a pair of random variables. Using these mechanisms, we will then cover some methods that can estimate the distribution of the longest path length. We classify existing methods into four categories: heuristics that provide an approximation[3]; methods that provide bounds; exact methods (that are not covered because of their time complexity); and the Monte Carlo approach. Last we describe the related application fields of this work.

3.1 Evaluation of Arithmetic Operations

3.1.1 Numerical Evaluation in the Case of Independent Random Variables

Characterizing the probability density function of the maximum or the sum of two *independent* random variables (also called *operands*) can be done using basic results from the probability theory [30].

Maximum of Two Independent Random Variables Let η and ε be two independent random variables. We call $\omega = \max(\eta, \varepsilon)$ the maximum of η and ε. The value of ω is lower than a constant z if and only if both operands are lower than z. Thus, the cumulative density function of

[3]As shown by Hagstrom [17], the problem is #P-hard in the discrete case.

the maximum of two independent random variables is the product of their cumulative distribution functions:

$$F_\omega(z) = F_\eta(z) \times F_\varepsilon(z) \tag{1}$$

When operands are discretized, methods from numerical analysis can be used to estimate the result (resampling, interpolation, etc.) [19]. Remark that determining the probability density function from a cumulative distribution function requires a numerical derivation. As derivations are numerically challenging, we often prefer to obtain the probability density function of ω. To this end, we analytically derive Eq. (1): $f_\omega(z) = F_\eta(z) \times f_\varepsilon(z) + f_\eta(z) \times F_\varepsilon(z)$. This formula requires the cumulative distribution functions of η and ε, which can easily be obtained by numerically integrating the probability density functions.

Sum of Two Independent Random Variables Consider the same operands in the sum $\omega = \eta + \varepsilon$. For discrete random variables, we have: $\Pr[\omega = z] = \sum_x \Pr[\eta = x] \times \Pr[\varepsilon = z - x]$. For continuous random variables, the probability density function of the sum of two independent random variables is the convolution of their probability density functions: $f_\omega = \int_x f_\eta(x) f_\varepsilon(z-x)dx = f_\eta * f_\varepsilon$.

The complexity of directly computing a convolution is $O(N^2)$ where N is the number of values representing a probability density function. Numerically, we can use the Fast Fourier Transform, whose complexity is $O(N \log N)$ to speed up this computation. Indeed, in the frequency domain, convolution is a product and its time complexity is linear.

3.1.2 Expected Value and Variance in the Case of Normal Distributions

When an operation is performed on random variables that are normally distributed, then the expected value and the variance of the result can be formulated in closed form even when the operands are dependent.

Maximum of Correlated Normal Laws Clark [11] proposed a set of formulas to cope with the maximum operation. These formulas characterize the first four moments of the maximum of two normal laws. Let η and ε be two random variables that follow each a normal law. Their expected values and variances are noted μ_η, μ_ε, σ_η^2 and σ_ε^2. The linear correlation coefficient between η and ε is $\rho_{\eta,\varepsilon}$. We define two functions: $\varphi(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$ and $\Phi(x) = \int_{-\infty}^{x} \varphi(t)dt$. Clark characterizes the expected value and the variance of $\omega = \max(\eta, \varepsilon)$, namely μ_ω and σ_ω^2:

$$\mu_\omega = \mu_\eta \Phi(b) + \mu_\varepsilon \Phi(-b) + a\varphi(b) \tag{2}$$

$$\sigma_\omega^2 = (\mu_\eta^2+\sigma_\eta^2)\Phi(b)+(\mu_\varepsilon^2+\sigma_\varepsilon^2)\Phi(-b)+(\mu_\eta+\mu_\varepsilon)a\varphi(b)-\mu_\omega^2 \tag{3}$$

where $a = \sqrt{\sigma_\eta^2 + \sigma_\varepsilon^2 - 2\sigma_\eta\sigma_\varepsilon\rho_{\eta,\varepsilon}}$ and $b = \frac{\mu_\eta - \mu_\varepsilon}{a}$.

Moreover, Clark provides a formula to compute the linear correlation coefficient between the result of a maximum and a given random variable τ:

$$\rho_{\tau,\omega} = \frac{\sigma_\eta \rho_{\tau,\eta}\Phi(b) + \sigma_\varepsilon \rho_{\tau,\varepsilon}\Phi(-b)}{\sigma_\omega} \tag{4}$$

Sum of Correlated Normal Laws Let us consider the sum $\omega = \eta + \varepsilon$. The following formulas are general results from probability theory. They are valid for any probability law that η and ε may follow.

$$\mu_\omega = \mu_\eta + \mu_\varepsilon \tag{5}$$

$$\sigma_\omega^2 = \sigma_\eta^2 + 2\sigma_\eta\sigma_\varepsilon\rho_{\eta,\varepsilon} + \sigma_\varepsilon^2 \tag{6}$$

$$\rho_{\tau,\omega} = \frac{\sigma_\eta\rho_{\tau,\eta} + \sigma_\varepsilon\rho_{\tau,\varepsilon}}{\sigma_\omega} \tag{7}$$

3.2 Heuristic Approaches

Heuristic methods often approximate the inputs to provide an estimation of the result. We classify them into three categories: approaches based on series-parallel reductions; methods based on the normality assumption; and the canonical approach.

3.2.1 Series-Parallel Reductions

A method based on a succession of reductions was first presented by Martin [27], and then by Dodin [14]. It provides an exact solution when the graph is series-parallel. Moreover, this method has a polynomial-time complexity.

It uses two kinds of reductions. We describe them by considering that all vertex weights are zero:

Series reduction If a vertex has exactly one incoming edge and one outgoing edge, then a series reduction is performed. It consists in eliminating the vertex and in replacing both edges by a single one whose weight is the sum of the weights of the initial edges. As the added weights are independent random variables, all techniques presented in Section 3.1 can be applied.

Parallel reduction A parallel reduction is performed if there exist two edges that share the same source and the same target. They are thus replaced by a single edge whose weight is the maximum of the weights of the initial edges. Again, operands are independent and we can use the techniques presented above.

For a given instance, all the possible reductions are performed. As one reduction may enable new reductions, they are performed iteratively until no more reduction is possible. The process ends up with a single edge between the source vertex and the sink if and only if the initial graph is series-parallel. The exact distribution of the longest path length is then given by the weight of the final edge.

If the graph is not series-parallel, then the process gives an irreducible graph containing several edges. It is still possible to continue the reductions by adapting the graph. A vertex is thus selected randomly among the ones that contain only one incoming edge. This vertex and its incoming edge are then duplicated multiple times in such a way that each new vertex is connected to exactly one of the outgoing edges of the initial vertex. If there is no vertex with only one incoming edge, then a symmetrical mechanism is performed for a vertex with only one outgoing edge. After this duplication step, all the enabled reductions are thus performed until a new irreducible graph is obtained. These two steps (reduction and duplication) are repeated until a single edge remains.

As some edges are duplicated when the graph is not series-parallel, the corresponding weights are also duplicated. This means that at each given step, the graph may contain weights that are not independent. Maximums and sums on dependent random variables raise issues (except if both operands follow normal laws). Thus, the result is generally not exact.

Bein et al. [3] improved this method by minimizing the number of duplicated vertices. Moreover, Ludwig et al. [25] perfected the approach by decreasing the algorithmic time complexity necessary to find new enabled series-parallel reductions.

3.2.2 Normality Assumption

Assuming that all the weights in a graph follow normal laws is common in the literature. The normality assumption concerns both intermediate results and the final distribution. This is a perfect use-case for Clark's formulas [11] that estimate the first four moments of the maximum of two normals (see Section 3.1.2).

This assumption is supported by the *central-limit theorem* that states that the sum of independent random variables tends to be normally distributed as the number of variables increases. As a graph encodes an arithmetic expression that may contain many additions, the result tends to approach a normal law if maximum operations do not significantly impact the resulting distribution.

The method proposed by Sculli [37] is a direct application of Clark's approach. Each random variable is reduced to its expected value and variance. Maximums are computed by considering that operands follow independent normal laws. The obtained result is again approximated as a normal law and its first two moments are computed with Clark's formulas.

Sculli's approach has, however, some limits. First, correlation coefficients between operands are always considered to be zero. This is false when operands relate to edges that have a common ancestor (see Section 2.5). Ignoring the effect coming from path reconvergence leads to an accumulation of errors that can be significant when the graph is large. In this paper, we propose two methods that are based on the same principle, but with techniques that estimate correlation coefficients.

The second limit is related to the normality assumption. Although input random variables are not normal in the general case, the assumption does not hold either when all weights are normal because the result of each maximum is approximated by a normal law.

Nevertheless, the normality assumption offers several advantages: we can use formal probabilistic results (Clark's formulas); the error is low as we will show in our experiments in Section 5; and, the algorithmic time complexities of methods based on this assumption are generally low.

To conclude, the relevance of this assumption depends on several criteria: the normality of input random variables; the depth of the graph, which determines the number of sums; and, the dependence and the similarity between the operands of each maximum, which determines the normality of intermediate results.

3.2.3 Canonical Representation

Evaluating the distribution of the longest path length is also required when designing digital circuit. Although we consider that all random variables in $\mathcal{X}$ are independent, proposed methods in this field are specifically designed to tackle spatial correlations, namely dependencies between the weights. For instance, Sapatnekar et al. [35] described how to apply principal component analysis to deal with these correlations. Spatial correlations make the problem more difficult.

With the canonical representation [43] that appeared in this context, dependencies between maximum operands (and spatial correlations) are efficiently taken into account. An extension, proposed by Zhang [48], improves the method and reduces its algorithmic time complexity.

In the canonical approach, each random variable (weights and intermediate results) are expressed using an expected value and a weighted sum of standard normal laws:

$$\eta = \mu + \sum_i \alpha_i \Upsilon_i$$

where μ is the expected value of η. Each random variable $\Upsilon_i \sim \mathcal{N}(0,1)$ follows a standard normal law (with mean $\mu = 0$ and variance $\sigma = 1$). Parameters α_i determine thus the variance of η. In this representation, all the normal laws Υ_i are independent. This is used for the dependencies between the weights.

Evaluating arithmetic operations on random variables in canonical representation makes partial use of the formulas proposed by Clark (see Section 3.1.2). Let $\eta = \mu_\eta + \sum_i \alpha_{\eta,i}\Upsilon_i$ and $\varepsilon = \mu_\varepsilon + \sum_i \alpha_{\varepsilon,i}\Upsilon_i$ be two random variables in canonical representation. The sum $\omega = \eta + \varepsilon$ can be evaluated as follows:

$$\omega = (\mu_\eta + \mu_\varepsilon) + \sum_i (\alpha_{\eta,i} + \alpha_{\varepsilon,i})\Upsilon_i$$

The maximum is defined as $\omega = \max(\eta, \varepsilon)$. Recall from Section 3.1.2 that $\Phi(x) = \int_{-\infty}^{x} \varphi(t)dt$, $a = \sqrt{\sigma_\eta^2 + \sigma_\varepsilon^2 - 2\sigma_\eta\sigma_\varepsilon\rho_{\eta,\varepsilon}}$ and $b = \frac{\mu_\eta - \mu_\varepsilon}{a}$.

The probability that η takes a value greater than ε, i.e. $\Pr[\eta > \varepsilon]$, is $\Phi(b)$. The maximum is approximated by:

$$\begin{aligned}\hat{\omega} &= \Phi(b)\eta + \Phi(-b)\varepsilon \\ &= (\Phi(b)\mu_\eta + \Phi(-b)\mu_\varepsilon) + \sum_i (\Phi(b)\alpha_{\eta,i} + \Phi(-b)\alpha_{\varepsilon,i})\Upsilon_i\end{aligned}$$

This evaluation of the maximum requires the correlation coefficient between the operands (i.e.,

$\rho_{\eta,\varepsilon}$):

$$\rho_{\eta,\varepsilon} = \frac{\sum_i \alpha_{\eta,i}\alpha_{\varepsilon,i}}{\sqrt{\sum_i \alpha_{\eta,i}^2}\sqrt{\sum_i \alpha_{\varepsilon,i}^2}}$$

The canonical approach relies on the normality assumption that is described above. Representing each random variable using a linear combination of standard normal laws provides an elegant and efficient method for characterizing the dependencies between each intermediate result. However, this is done to the detriment of the maximum operation whose precision is worse than with Clark's approach. Indeed, whereas Clark's approach provides the exact first four moments of the maximum of two normals, the canonical approach approximates the maximum as a linear combination of normals, which is inexact even when operands actually follow normal laws.

3.3 Bounds

Several methods provide bounds on the distribution of the longest path length. We first define the first-order stochastic dominance [31, Definition 1.2.1] that we use to determine if two random variables are comparable and, if possible, to know which one is greater. Let η and ε be two random variables. We say that η dominates stochastically ε if $\Pr[\eta \leq x] \leq \Pr[\varepsilon \leq x]$ for all x.

Kleindorfer [21] has proved a lower and an upper bound on the distribution of the longest path length. The upper bound is given by assuming that all maximum operands are independent and, hence, by directly applying the mechanism of Section 3.1.1. For the lower bound, maximum operations are not executed. Instead, the distribution of one of the operands is selected as the result.

The approach using series-parallel reduction described in Section 3.2.1 also gives an upper bound by transforming any given graph into a series-parallel one. This result improves Kleindorfer's upper bound.

Yazici-Pekergin et al. [47] have proposed to replace NBUE (New Better Than Used in Expectation [4]) distributions in a graph by an upper bound. This technique is useful when we know only the expected value of the random variables and when they all verify the NBUE property.

Finally, some methods only propose a bound on the expected value of the result. Fulkerson [15] has proposed one of the first lower bound of the literature. It has been improved by Robillard [34] using Jensen's inequality. Kamburowski [20] proposed to bound the expected value and the variance using the normality assumption and Clark's formulas. Finally, Weiss [45] also gave bounds on related quantities such as the shortest path.

3.4 Monte Carlo Method

The Monte Carlo method proposed in this context [7, 42] consists in repeatedly transforming the random weights into deterministic ones.

For each random variable of the graph, a value is drawn according to its law. When this is done, we obtain a unique value for the graph. This step is repeated several times generating a new value at each iteration. The set of resulting values defines an empirical distribution function that approaches the resulting distribution as the number of iterations increases.

We need to define the number of trials (noted T) required to achieve a given precision. If we assume that the distribution of the longest path length follows a normal law, then Cochran's theorem states that the variance follows a χ^2 law with $T-1$ degree of freedom. Hence, the number of degrees of freedom needed to obtain a required confidence interval directly gives the number of iterations to perform. For instance, with 20,000 iterations, the relative error of the standard deviation is lower than 5% with a confidence level of 99%. With one million iterations, the error is lower than 1%.

[4]Intuitively, NBUE distributions are distributions that describe the remaining lifetime of objects and such that new objects have a better expected lifetime than used objects.

Another way of quantifying the error is given by the Kolmogorov-Smirnov statistic: it measures the distance between the empirical cumulative distribution function and the true distribution. According to the Kolmogorov distribution, this difference is lower than $1.629/\sqrt{T}$ with a confidence level of 99% when the number of iterations exceeds 100 [36]. For 20,000 iterations, the difference is lower than 1.2%. For one million iterations, it is lower than 1.629‰.

The Monte Carlo method has two advantages. First, the empirical distribution function converges towards the resulting distribution as $T \to \infty$ according to the Glivenko-Cantelli theorem. Second, this method is not sensible to operand dependencies when performing a maximum operation.

3.5 Related Research Fields

Evaluating the distribution of the longest path length of a DAG (Directed Acyclic Graph) with random weights arises in several fields:

- the problem was first defined by Malcolm et al. [26] in the context of project management. A project is assumed to be divided into a set of activities that are structured through a set of events. Each activity has to be performed by a resource and an event can be reached upon its completion. As activity durations can be modeled by random variables, the overall project consists of a graph where each edge corresponds to an activities and each vertex to an event;

- task graph scheduling on parallel machines with random durations (see Section 2.4) has then been introduced [10, 32, 44]. Several references are provided in [33];

- last, the problem appears when designing digital circuits. A digital circuit is a network of gates that are connected through wires. In order to predict the performance of such circuits, static timing analysis is performed to estimate the propagation delay of a signal from the input to the output gates. As variations may occur when manufacturing digital circuits, the propagation delay of each wire and each gate is uncertain. Analyzing a digital circuit requires thus to evaluate the distribution of the longest path length of a DAG where each edge corresponds to a wire and each vertex to a gate. We report the reader to the survey proposed by Blaauw et al. [5] for more details on this field.

3.6 On the Complexity of the Studied Problem

Although the problem is frequently mentioned to be #P-hard in the literature, there is sometimes a slight confusion on the precise problem that is considered.

In this paper, we consider the numerical problem of determining the distribution of the longest path length of a DAG with random weights. The random variables are assumed to follow continuous distributions (such as the uniform distribution) and the output of the problem is the probability that the longest path length takes a value lower than or equal to some given constant. The problem consists in approximating this probability to some given number of correct digits.

On the other hand, the problem that is known to be #P-hard [17] is when random variables are discrete with only rational values. In this case, the objective is to find the exact probability, which is rational, that the longest path length takes a value lower than or equal to some given constant.

Although we may infer that the problem with discrete random variables (possibly with irrational values) is also difficult, we cannot conclude on the complexity of the continuous version of the problem. However, we suspect that this version is also difficult as the challenge when directly evaluating the solution is similar in both versions: the number of longest paths may be exponential.

4 Proposed Methods

Although many bounds have been proposed, either they do not provide estimations that are precise enough in practice or their time complexity is prohibitive. We propose two practical heuristics that are based on the normality assumption presented in Section 3.2.2 and on Clark's formulas described in Section 3.1.2. Namely, we approximate each random variable and each result of a maximum as a random variable that follows a normal law. Both our methods improve Sculli's approach [37] because they use a mechanism to estimate correlations between maximum operands whereas Sculli's approach does not.

4.1 CorLCA

The first described heuristic is called *CorLCA* (Correlation based on Lowest Common Ancestor). This method visits each vertex of a graph only once. For each vertex, correlations between the operands of a maximum operation are estimated using an efficient method. The objective of this method is to offer precise results without significantly increasing the algorithmic time complexity compared to Sculli's approach (presented in Section 3.2.2). To this end, the correlation between each pair of maximum operands is estimated by determining their lowest common ancestor. Algorithm 1 presents the steps of CorLCA.

First, we describe the general behavior of the algorithm and we detail the construction of a tree (called *correlation tree* below) that allows efficient searches for the lowest common ancestor of any pair of vertices. Then, we show how to compute a correlation coefficient using this ancestor. Finally, we analyze the complexity of CorLCA.

CorLCA relies on a main loop that visits vertices of a graph $G(V, E, \mathcal{X})$ in a topological order.

For each iteration, two types of operations are performed:

- intermediate result evaluations (Lines 4, 9, 10, 14 and 20)

ALGORITHM 1: Heuristic CorLCA based on lowest common ancestor queries to estimate the correlation between two operands

```
Require: $G = (V, E, \mathcal{X})$ {Directed acyclic graph with random weights}
Ensure: $(\mu, \sigma^2)$ {Estimation of the expected value and variance of the distribution of the longest path length of $G$}
1:  for $i = 1$ to $n$ do {Visit all the vertices in a topological order}
2:    $\dot{v}_i = 0$ {Initialization of $v_i$ parent in the correlation tree}
3:    for all $v_j \in \text{Pred}(v_i)$ do {Visit all the predecessors of $v_i$}
4:      $Y_{ji} = X_{ji} + Y_j$ {Equations 5 and 6}
5:      if $\dot{v}_i = 0$ then {First iteration of the loop}
6:        $\dot{v}_i = v_j$
7:        $\eta = Y_{ji}$
8:      else
9:        $v_k = \text{LCA}(\dot{v}_i, v_j)$ {Determine the Lowest Common Ancestor of $\dot{v}_i$ and $v_j$}
10:       $\rho_{\eta, Y_{ji}} = \frac{\sigma^2_{Y_k}}{\sigma_\eta \sigma_{Y_{ji}}}$ {Estimate the correlation between $\eta$ and $Y_{ji}$}
11:       if $\Pr[\eta < Y_{ji}] > 0.5$ then {If vertex $v_j$ is preponderant in the maximum}
12:         $\dot{v}_i = v_j$ {Change the predecessor of $v_i$ in the correlation tree}
13:       end if
14:       $\eta = \max(\eta, Y_{ji})$ {Equations 2 and 3, and Line 10}
15:     end if
16:   end for
17:   if $\dot{v}_i = 0$ then {Vertex $v_i$ has no predecessor}
18:     $Y_i = X_i$
19:   else
20:     $Y_i = X_i + \eta$ {Equations 5 and 6}
21:   end if
22: end for
23: return $(\mu_{Y_n}, \sigma^2_{Y_n})$
```

- incremental construction of the *correlation tree* (Lines 11 to 13)

This last tree is rooted and is used only for computing correlations between maximum operands. It contains the same vertices as graph G and a subset of its edges. In particular, each vertex in the correlation tree has only one incoming edge with the exception of the root, which has none. At each iteration of CorLCA, the predecessor $\dot{v}_i$ of the visited vertex v_i is retained as the unique parent of v_i in the correlation tree.

4.1.1 Correlation Tree Construction

Selecting a unique parent for a given vertex in the correlation tree is done by determining which predecessor has the most significant impact on the maximum operation. This means that we want to select the edge that influences the most the intermediate result of a maximum. Thus, the correlation tree approximates the structure of the correlations between each pair of intermediate results.

When a vertex contains several incoming edges, the selected edge is the one whose intermediate result is greater than the intermediate results of the other incoming edges with the highest probability. Let $Y_{ji} \sim \mathcal{N}(\mu_{Y_{ji}}, \sigma_{Y_{ji}})$ and $Y_{j'i} \sim \mathcal{N}(\mu_{Y_{j'i}}, \sigma_{Y_{j'i}})$ be two normal random variables representing the intermediate results of two edges targeting the same vertex v_i. In Section 3.1.2, we defined function $\Phi(x) = \int_{-\infty}^{x} \varphi(t)dt$ and symbol b. Section 3.2.3 mentions that the probability of Y_{ji} being greater than $Y_{j'i}$ is $\Pr[Y_{ji} > Y_{j'i}] = \Phi(b)$. This mechanism is used to select each predecessor (Lines 11 to 13). In the case of normal distributions, it is actually sufficient to compare only the expected values of Y_{ji} and $Y_{j'i}$ to determine which one is greater with the highest probability (the one with the highest expected value is selected).

A complete correlation tree that could correspond to the graph of Figure 2 is represented on Figure 4. For each vertex that has several incoming edges, a single edge is selected.

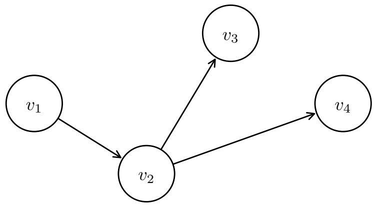


Figure 4: Possible correlation tree corresponding to the graph of Figure 2.

4.1.2 Estimation of Correlation Coefficients

Equations 2 and 3 of Section 3.1.2 are used at Line 14 to compute the maximum of two random variables and they require the correlation of the operands beforehand (which is done on Line 10).

The correlation tree enables an efficient estimation of the correlation of any pair of intermediate results. By finding the lowest common ancestor of two vertices, it is possible to compute directly the correlation between the intermediate results of these two vertices. Let Y_{ji} and $Y_{j'i}$ be two intermediate results of two edges targeting the same vertex v_i. Let v_k be the lowest common ancestor between vertices v_j and $v_{j'}$ in the correlation tree. Its intermediate result is noted Y_k. Our approximation consists in considering that $Y_{ji} = \eta + Y_k$ and $Y_{j'i} = \varepsilon + Y_k$ where η and ε are two random variables independent of Y_k. Random variables η and ε are independent because they represent the sums of the weights on the paths between vertex v_k and vertices v_j and $v_{j'}$, respectively (these paths do not share any vertex by definition of the lowest common ancestor). Hence, the correlation between Y_{ji} and $Y_{j'i}$ is:

$$\rho_{Y_{ji},Y_{j'i}} = \frac{\sigma_{Y_k}^2}{\sigma_{Y_{ji}}\sigma_{Y_{j'i}}}$$

It is, however, an approximation because vertices v_j and $v_{j'}$ can have several lowest common ancestors in the complete directed acyclic graph.

This mechanism is similar to the second optimization of the method proposed by Yao et al. [46]. However, our method is finer in case of multiple lowest common ancestors.

4.1.3 Complexity

The time complexity of CorLCA depends on the cost of the method used to find the lowest common ancestor of two vertices in a tree in which vertices are inserted incrementally. Let λ (resp., ν) be the time (resp., space) complexity necessary to insert the vertices and to perform LCA (Lowest Common Ancestor) queries. Then, the time complexity of CorLCA is $O(m\lambda)$ and its space complexity is $O(n + \nu)$.

Cole et al. [12] have presented a method that performs vertex insertions and LCA queries in constant time if insertions do not double the size of the tree. As this assumption does not hold in our case, data structures would require to be rebuilt periodically with this method. Moreover, their approach tackles vertex insertions inside the tree and vertex removals, which CorLCA does not need. Gabow [16] has described an algorithm that performs leaf insertion in amortized constant time.

The problem consists in alternating LCA queries and leaf insertions in the same tree. To the best of our knowledge, the literature does not provide an optimal algorithm for this specific problem. Given the related works presented above, we conjecture that $\lambda = O(1)$ and $\nu = O(n)$, which would lead to a time complexity of $O(m)$ and a space complexity of $O(n)$ for CorLCA.

4.2 Cordyn

This second heuristic, called *Cordyn* (Correlation based on a dynamic programming approach), takes into account dependencies caused by reconvergent paths. A dynamic programming approach is used to determine the correlation coefficients that are required when applying Clark's formulas. Despite a higher time complexity than with CorLCA, estimated correlation coefficients are more precise. Indeed, no approximation other than the normality assumption is done.

4.2.1 Algorithm

The algorithmic principle lies in continuously characterizing the set of correlation coefficients that could be required when a maximum is performed with Equations 2 and 3. Although it is always possible to determine recursively any correlation coefficient (with a recursive method using Equation 4), some of the computed coefficients are used multiple times and it is suboptimal to recompute them. As the problem raised by the determination of these coefficients exhibits sub-problems that overlap, we propose a dynamic programming strategy. Then, for each newly visited vertex v_i, all the correlation coefficients ρ_{Y_i,Y_j} are computed and kept in a symmetric square matrix $P = (\rho_{Y_i,Y_j})_{1\leq i\leq n, 1\leq j\leq n}$ of size $n \times n$.

Algorithm 2 describes the main loop of Cordyn, which visits vertices in a topological order. Intermediate result evaluation is done on Lines 3, 7 and 8 by reusing newly computed correlation coefficients (Line 10). Lines 4, 9 and 10 are used to compute correlation coefficients between a given random variable and each intermediate result that corresponds to any already visited vertex (i.e., any of the random variables in the set $\{Y_k : 1 \leq k < i\}$ where v_i is the visited vertex in the current loop iteration). Coefficients computed on Line 4 are only required for the computations on Lines 5 and 9 that are only used themselves on Lines 7 and 10, respectively. However, correlations determined on Line 10 can be used during future iterations of the main loop on Line 4 and this is why matrix P is updated with the obtained values.

We finish the description of Cordyn with two remarks. When the number of incoming edges of v_i is strictly greater than two, the operands on Line 7 are grouped pairwise and Clark's formulas are used successively (coefficient computations must then be adapted on Line 9). The second remark is related to the fact that an in-

ALGORITHM 2: Heuristic Cordyn based on a dynamic programming approach to determine the correlation between two operands

```
Require: $G = (V, E, \mathcal{X})$ {Directed acyclic graph with random weights}
Ensure: $(\mu, \sigma^2)$ {Estimation of the expected value and variance of the distribution of the longest path length of $G$}
1:  for $i = 1$ to $n$ do {Visit all the vertices in a topological order}
2:      for all $v_j \in \text{Pred}(v_i)$ do {Visit all the predecessors of $v_i$ by increasing indices}
3:          $Y_{ji} = X_{ji} + Y_j$ {Equation 5 and 6}
4:          compute $(\rho_{Y_{ji},Y_k})_{1 \leq k < i}$ {Equation 7 and matrix $P$}
5:          compute $(\rho_{Y_{ji},Y_{j'i}})_{v_{j'} \in \text{Pred}(v_i), j' < j}$ {Equation 7, matrix $P$ and Line 4}
6:      end for
7:      $\eta = \max_{v_j \in \text{Pred}(v_i)}(Y_{ji})$ {Equations 2 and 3, and Line 5}
8:      $Y_i = X_i + \eta$ {Equations 5 and 6}
9:      compute $(\rho_{\eta,Y_k})_{1 \leq k < i}$ {Equation 4 and Line 4}
10:     compute $(\rho_{Y_i,Y_k})_{1 \leq k < i}$ and complete $P$ {Equation 7 and Line 9}
11: end for
12: return $(\mu_{Y_n}, \sigma^2_{Y_n})$
```

termediate result can be discarded when computing correlations as soon as every successors of the corresponding vertex have been visited. Indeed, on Lines 4, 9 and 10, we can reduce the set of considered intermediate results to those corresponding to the vertices $\varphi(v_i)$, where $\varphi(v_i)$ is the set of vertices v_j such that $j < i$ and such that $\exists(v_j, v_k) \in E, i < k$. Thus, the topological order in which vertices are visited has an impact on the efficiency (temporal and spatial) of the method, but no influence on the result quality (except for round-off errors).

4.2.2 Complexity

To determine the complexity of Cordyn, we introduce and remind some notations: let $\deg^-(v)$ be the number of incoming edges of vertex v, $n = |V|$ the number of vertices and $m = |E|$ the number of edges. The most costly step consists in characterizing correlations between each new vertex intermediate result and all the obtained intermediate results (Line 4). Remember that the vertices are visited by increasing indices. At step i, determining correlation coefficients between each obtained intermediate result $(Y_k)_{1 \leq k < i}$ and Y_{ji} costs $O(i)$ operations and is repeated $\deg^-(v_i)$ times for each predecessor of v_i. The time complexity of the approach is thus $O(\sum_{i=1}^{n}(i \times \deg^-(v_i))) = O(nm)$. Moreover, $O(n^2)$ elements must be stored in matrix P.

4.3 Canonical Representation Adaptation

We also adapted the canonical method (see Section 3.2.3) to our problem. In the scheduling problem, weights are independent, which means that each weight corresponds to a single random variable Υ with the canonical representation. We note each weight $\hat{X}_i$ because the canonical representation is an approximation of the true random variable X_i. By considering that all edges have zero weights, then $\hat{X}_i = \mu_i + \alpha_i \Upsilon_i$ and $\hat{Y}_j = \mu_j + \sum_i \alpha_{ji} \Upsilon_i$. This last equation means that each intermediate result can be expressed as a linear combination of all standard normal distributions that correspond to the weights of the graph. As an intermediate result Y_j can be the result of a maximum (which does not produce a linear combination of normal laws), it is clear that the last equation is an approximation.

4.4 Unfavorable Example

Figure 5 illustrates all the steps when considering a graph with four vertices. Edge weights are zero and each vertex weight follows an exponential law with an expected value of one. As exponential laws differ significantly from normal laws, this example is not favorable to approaches based on the normality assumption. However, CorLCA improves Sculli's approach, which ignores dependencies between intermediate results Y_2 and Y_3. Both these random variables are operands of the maximum performed to compute the final result Y_4. For this graph, CorLCA and Cordyn produce the same result because the graph structure is simple and no pair of vertices contains more than one lowest common ancestor.

5 Empirical Validation

5.1 Instances

To empirically validate our methods, we use a set of instances based on existing testbeds that define the graph structure but have either deterministic weights or no weight. Each weight is replaced by a random variable whose expected value is equal to the deterministic weight. We will describe how to determine the distribution of each random variable.

The graph structures come from the following three sets of instances:

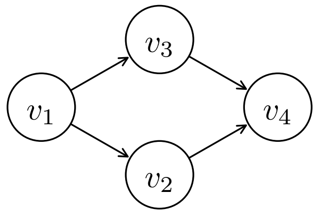
- RCPSP instances of the PSPLIB Library [22] in the case of project management. Graphs are classified according to 4 sizes: 30, 60, 90 and 120 vertices. There are 48 instances in each of the first three classes and 60 in the last one (we use the first variant of each of these instances).
- The STG [41] set for task graphs. For each distinct size (from 50 to 1500 vertices), we select 24 instances, each using a distinct generation method. Each generator results from a combination of one of the four graph generators and one of the six cost generators (we also use the first variant of each combination).
- ISCAS-85 [18] and ISCAS'89 [6] for digital circuits (45 instances).

The expected value of each weight is not fully determined in the case of digital circuits because they do not have weight. The expected value is arbitrarily set to 1.

The structure of the graph and the expected value of each weight is specified as shown above. We need to transform this into a probabilistic instance by determining the variance and the law of each weight.

First, we need to specify the variance of each weight. Instead of managing variances, we use the coefficient of variation (CV), which measures the relative dispersion of the law. Formally, it is the ratio between the standard deviation and the mean of a set of values. For each weight, we draw the CV according to a gamma law such that the standard deviation of all CV is 10% of their expected value. This introduces some heterogeneity in the weight CV (otherwise, expected values would be proportional to standard deviations and this could bias the evaluation). However, when the standard deviation of a CV is too high, then some CV have a high value. As a compensation, most other CV have to be close to zero (as CV must be non-negative) and can thus be considered deterministic (which could again bias the evaluation). The 10% value used for the heterogeneity in the weight CV is therefore a compromise.

Second, we decided that, for a given graph, all weights would follow the same law (among the five given in Table 2) but with different parameters. These parameters may be inferred from the expected value and chosen variance of the



Intermediate results	Sculli [37]		CorLCA		Formal analysis	
	μ	σ^2	μ	σ^2	μ	σ^2
Y_1	1	1	1	1	1	1
Y_2	2	2	2	2	2	2
Y_3	2	2	2	2	2	2
Y_4	3.80	2.36	3.56	2.68	7/2	13/4

Figure 5: Intermediate results in a graph with four vertices whose weights follow exponential laws with expected value one. CorLCA and Cordyn give the same result for this graph. In this simple example, the density function of the result can be formally derived and is $(x^2 - 2x + 2 - 2e^{-x})e^{-x}$.

weight. This is direct for the uniform and normal laws. The exponential law has only one parameter that is determined by the expected value (the variance is thus discarded). If weights follow a beta law, then the maximum and minimum values are determined by the expected value and the variance while the shape parameters are $\alpha = 2$ and $\beta = 5$. For the Weibull law, we used a simple dichotomic iterative method to set the scale and shape parameters.

The $48 \times 3 + 60 = 204$ PSPLIB instances, the $24 \times 8 = 192$ STG instances and the 45 ISCAS instances are generated with the default values given in Table 2. Moreover, a subset of these instances (48 PSPLIB, 24 STG and all ISCAS) are used with the tested values. For each tested value, the other parameter takes its default value. This leads to a total of $(204 + 192 + 45) + (48 + 24 + 45) \times 7 = 1260$ graphs.

Table 2: Parameters and values for the instance generation.

Parameter	Default value	Tested values
Law	uniform	normal, exponential, beta, Weibull
Expected value of the coefficient of variation	0.1	0.01, 0.03, 0.3

5.2 Method Qualities

To evaluate the quality of CorLCA and Cordyn, we compare them to three other methods: Sculli's approach (Section 3.2.2), a series-parallel reduction approach using numerical methods (Section 3.2.1) and the canonical approach (Section 3.2.3).

All the comparisons are done using *Emapse* (Evaluation of MAx-Plus Stochastic Expression), a tool described in Appendix 7.

In the following, the results obtained with the Monte Carlo method with at least one million iterations serve as the reference. Among the possible error metrics for assessing the solutions, we have chosen the Kolmogorov-Smirnov statistic as explained in Appendix 8.

5.2.1 General Comparison

Figure 6 presents a summary of the efficiencies of the five compared methods. For each generated instance and for each pair of methods (characterized by a line and a column), we compute the ratio between the Kolmogorov-Smirnov statistics for both methods (ratio of the line over the column). If this ratio is equal to one, then both methods produce an identical result in terms of precision. If this ratio is lower than 1, then the result of the method on the line is better than the method on the column. Each plot in the lower-left part shows the histogram and the ECDF of these ratios. The upper-right part summaries these data (each summary corresponds to the ra-

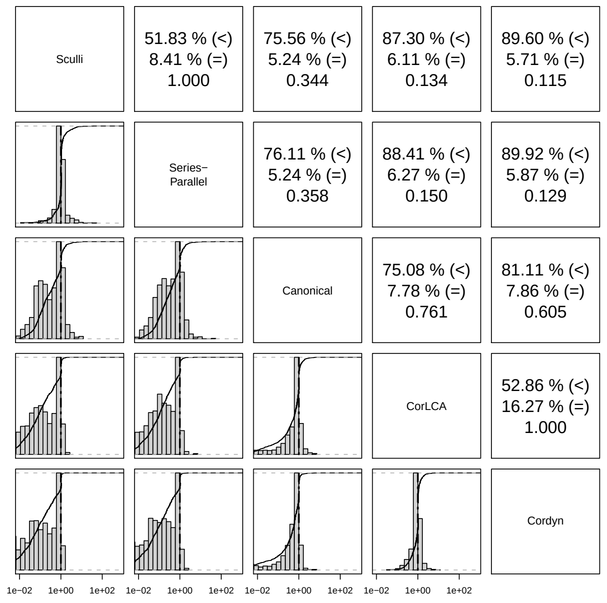


Figure 6: Pairwise comparison of five methods on 1 260 graphs using the Kolmogorov-Smirnov statistic as the error metric. Lower-left part: histograms and ECDF of the ratios between error measures of each pair of methods (ratio of the row to the column). Upper-right part: fraction of the ratios that are greater or equal to 1 and median ratio.

tios shown in the symmetric cell). For instance, we remark that the results produced by the canonical approach are less precise than those produced by CorLCA in 75.08% of the cases and that they have the same precision in 7.78% of the cases. Finally, the median ratio is 0.761 in this example.

We can conclude that both the proposed methods behave favorably relatively to the other three methods. We also remark that Sculli's approach is the worst and that the method based on series-parallel reductions is not far. Cordyn is the best heuristic and is slightly better than CorLCA. The canonical approach is at the middle: it is significantly better that the worst two approaches and significantly less precise than the proposed heuristics.

5.2.2 Comparison to An Upper Bound

We recall that the series-parallel reduction approach produces an upper bound on the distribution of the longest path length (see Section 3.3). We observe that our approaches (CorLCA and Cordyn) are more precise than this upper bound in more than 88% of the cases. This supports our motivation for precise heuristics even if no guarantee is provided. Indeed, in this case, an approximated algorithm (that gives an upper bound) can be much farther from the correct value than a heuristic.

5.2.3 Parameter Effects

We show the effect of each parameter on the precision of each method on Figure 7 to Figure 9. We focus on three parameters: the number of weights in a graph, the laws followed by these weights and the expected value of the coefficients of variation of these weights.

The precision of the five considered methods are assessed with the Kolmogorov-Smirnov statistic (as in the previous section) and are represented through boxplots. A boxplot consists of a five number summary of a set of values: the median is the thick horizontal segment, the box extends from the first quartile to the third one, and the length of the whiskers is 1.5 times the interquartile range. Note that with one million iterations, we are 99% confident that the difference of the reference (given by the Monte Carlo method) with the exact distribution is below 1.629‰ (see Section 3.4). Therefore, any statistic that is below this threshold (depicted by a horizontal line on the figures) means that the true error lies between 0 and $2 \times 1.629‰$. On this opposite, values are upper bounded by 1.

On Figure 7, we see how varies the precision of each method when the number of random variables increases. The precision of Sculli's and the series-parallel reduction approaches decrease when the size of the graph increases. The median of the Kolmogorov-Smirnov is always greater than 0.2 when the size is greater than or equal

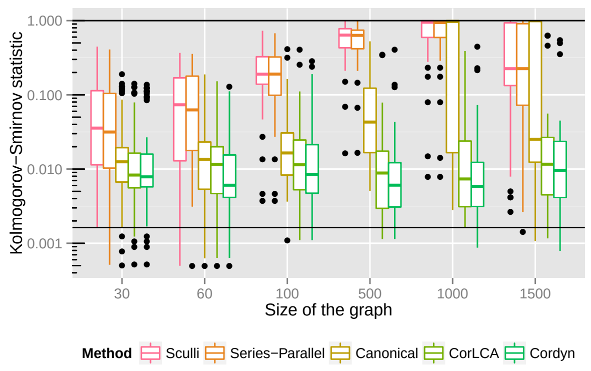


Figure 7: Effect of the number of random variables on the precision.

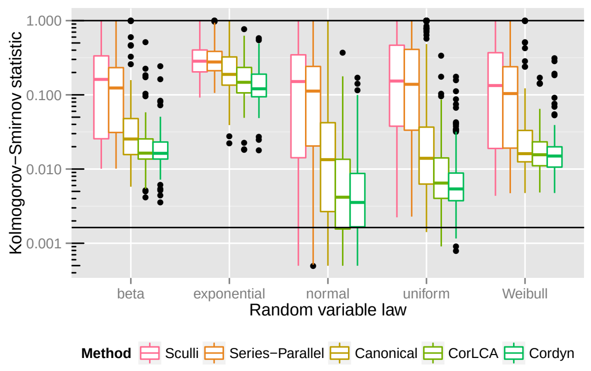


Figure 8: Effect of the law on the precision.

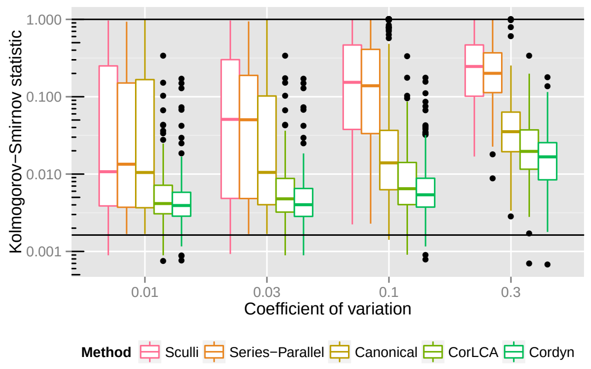


Figure 9: Effect of the expected value of the coefficient of variation of all weights on the precision.

to 500, whereas this value is lower for smaller graphs. There is indeed a significant change when the size reaches this value. The canonical approach has a similar behavior, though the precision of this method is globally better. In contrast, the precision of CorLCA and Cordyn are relatively stable when the size of the graph increases.

All approaches against which we confront our methods are considerably imprecise for graphs containing more than a thousand random variables: for the three approaches, the medians of the Kolmogorov-Smirnov statistics are close to one with 1 000 random variables.

The effect of the probability law that is selected for the entire graph is shown on Figure 8. The exponential law poses the greatest difficulty (even for approaches relying on numerical methods). This is indeed the law that differs the most from the normal law among those that are tested. In addition, its discretization is difficult due to its steep slope at the origin and its long tail. This imposes to perform an antagonist choice between a short discretization interval and a large definition support. On the contrary, normal and uniform laws are the most favorable to our heuristics and to the canonical approach (the other two methods behave similarly for all laws distinct from the exponential one).

Last, Figure 9 depicts the effect of the coefficient of variation (ratio of the standard deviation to the expected value of each weight) on the precision. We remark a general trend for each method that consists in a loss of precision when this coefficient increases. We explain this by the simplicity with which a maximum is evaluated when operand supports do not overlap. In this case, the result of the operation is the operand with the highest expected value. The greater the coefficient of variations, the lower the number of situations for which this direct method can be applied. Thus, increasing the uncertainty in a directed acyclic graph with random weights accentuates the errors related to the maximums.

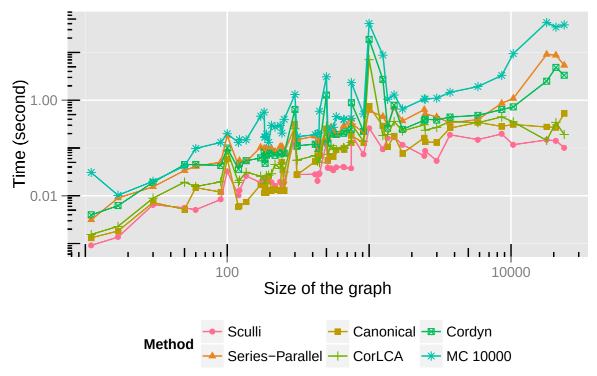


Figure 10: Computation times of each method in function of the number of random variables in the graph.

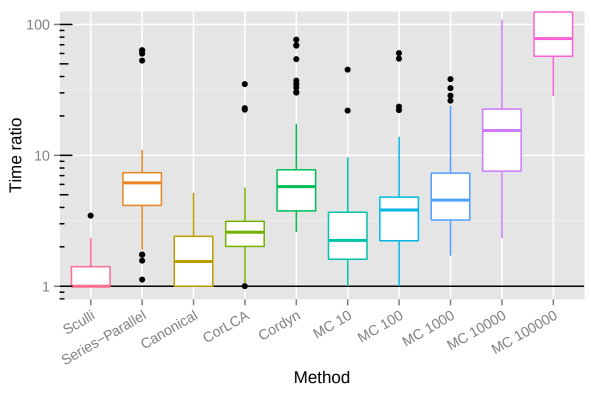


Figure 11: Ratios of the computation times of each method over the best computation times among all five methods and five Monte Carlo methods with varying number of iterations.

5.3 Computation Time

We study the computation times of several methods on the previous subset of 117 instances. The Monte Carlo method was also included with 10 to one hundred thousand iterations. To avoid unrelated initialization time, each method was run twice in the same program and only the second run was measured. A single core of an Intel(R) Xeon(R) CPU E5-2660 at 2.20GHz (SandyBridge) was used. More than 88% of the execution times are below 1 second with a median time of 67 ms. The average execution times are depicted on Figure 10. For clarity, only the execution time for the Monte Carlo method with ten thousand iterations is shown. With this number of iterations, we are 99% confident that the difference with the exact distribution is below 1.629% (see Section 3.4). Increasing the size of the instance has a clear effect on the performance of all methods. However, the execution time depends on other parameters than the number of random variables such as the structure of the graph, this is why the execution time is not strictly increasing with the size. For instance, there is no maximum in the case of chains, whereas there is as much maximums as sums in the case of in-trees.

Figure 11 presents ratios of computation times. For each instance, the computation time of each method is divided by the minimum computation time among all the methods. A ratio close to one indicates that the achieved performance is close to the best one. Note that the Monte Carlo method has execution ratios greater than 100 (in particular for the largest two number of iterations), which are not shown in this figure. Also, the computation times are summarized independently of the number of weights in the graphs because the impact of this parameter was found to be low.

Sculli's and the canonical approaches are the fastest methods. The canonical approach constitutes thus an interesting trade-off between the speed and the precision. CorLCA is the third fastest method (excluding Monte Carlo simulations). This figure indicates that Cordyn and the approach based on series-parallel reductions are comparable.

The ratios for the Monte Carlo simulations depend particularly on the implementation of the pseudo-random generator[5] and should be interpreted with caution. The previous analysis shows that the precision of Canonical, CorLCA

[5]A specialized coprocessor for generating pseudo-random values could significantly impact these results.

and Cordyn are mostly comparable to or lower than 1.629%, which is the precision achieved by the Monte Carlo method when the number of iterations is ten thousand iterations. Thus, our experiments suggest that these three methods outperform the Monte Carlo method given our hardware and software environment.

The complexity of CorLCA is conjectured to be the same as Sculli's approach. However, the implemented method performing LCA requests does not have a constant time complexity, which penalizes CorLCA.

In most cases, the series-parallel reduction approach is dominated by the canonical approach both in terms of speed and in terms of precision. This dominance is sufficiently important to conclude that series-parallel reductions are not an efficient mechanism in our context. The four other approaches (excluding Monte Carlo simulations) represent each a distinct compromise in terms of speed and precision.

6 Conclusion

In this paper, we study the problem of evaluating the distribution of the completion time of a set of dependent jobs with random durations. More precisely, the evaluation consists in characterizing the law that follows a random variable that is defined by a succession of maximums and sums of random variables. In practice, this problem difficulty comes from dependencies between the intermediate results, which is due to path reconvergences. In contrast, when sums and maximums are applied on independent random variables, then the evaluation is direct.

Two heuristics, CorLCA and Cordyn, are proposed. They represent each a new compromise in term of speed and precision relatively to the approaches against which our heuristics are confronted to. Moreover, we empirically compare approaches from the literature. Last, all these approaches have been implemented in a software, Emapse, which allows a fair comparison of several approaches in a unified way.

We show that series-parallel reductions do not constitute an interesting method for the considered instances as it is often less precise and slower than the canonical approach. CorLCA is the fastest heuristic among the ones for which the precision remains stable when the size of the graph increases. It is therefore a relevant choice when designing a scheduling heuristic that requires an efficient evaluation mechanism.

This paper focuses mainly on the scheduling problem, but our solutions can be applied to other related fields (project management and circuit design) as the models are similar. For instance, in the digital circuit case, there are dependencies between the random variables of the set $\mathcal{X}$. Although CorLCA ignores these dependencies, Cordyn can be generalized by considering correlations between weights in the graph. In this case, correlation computations must be extended to the correlation coefficients between the intermediate results (Y_i and Y_{ij}) and each weight $X \in \mathcal{X}$ (with the same asymptotic complexity).

One future work direction concerns the analytical study of the problem. There exist exact approaches when the graph structure verifies some specific properties and when all random variables follow the same type of laws. Using a law that is closed both under the maximum and the sum would provide an exact method for series-parallel graphs.

Another direction is about the empirical validation of the approaches. This could easily be extended to compare other approaches from the literature. Moreover, the factors that degrade the estimation of each approach (in particular for the canonical approach) remain to be determined.

Finally, yet another direction is to investigate the analysis of the algorithmic time complexity of both proposed. On one hand, characterizing the time complexity of CorLCA requires to study a problematic related to searching for the lowest common ancestor in a rooted tree. On the other hand, minimizing the time complexity of Cordyn using an optimal topological order is related to the vertex separation problem and the sum cut problem [13]. These problems, however, are beyond the scope of this paper.

Acknowledgments

Acknowledgment

We would like to thank Ihab El Alami for his participation to the development of the platform Emapse, in particular for his implementation of the series-parallel reduction approaches.

Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

References

[1] T. W. Anderson. On the distribution of the two-sample cramér-von mises criterion. *The Annals of Mathematical Statistics*, 33(3):1148–1159, September 1962.

[2] Alfredo H-S. Ang and Wilson H. Tang. *Probability Concepts in Engineering: Emphasis on Applications to Civil and Environmental Engineering.* Wiley, 2 edition, 2006.

[3] Wolfgang W. Bein, Jerzy Kamburowski, and Matthias F. M. Stallmann. Optimal reduction of two-terminal directed acyclic graphs. *SIAM Journal on Computing*, 21(6):1112–1129, 1992.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bulletin of the Calcutta Mathematical Society*, 35(4):99–109, 1943.

[5] David Blaauw, Kaviraj Chopra, Ashish Srivastava, and Lou Scheffer. Statistical timing analysis: From basic principles to state of the art. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 27(4):589–607, April 2008.

[6] Franc Brglez, David Bryan, and Krzysztof Koźmiński. Combinational profiles of sequential benchmark circuits. In *IEEE International Symposium on Circuits and Systems*, pages 1929–1934, 1989.

[7] John M. Burt and Mark B. Garman. Conditional Monte Carlo: A Simulation Technique for Stochastic Network Analysis. *Management Science*, 18(3):207–217, November 1971.

[8] Louis-Claude Canon. Emapse: code and scripts. May 2015.

[9] Louis-Claude Canon and Emmanuel Jeannot. Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments. *IEEE Transactions on Parallel and Distributed Systems*, 21(4):532–546, April 2010.

[10] K. M. Chandy and P. F. Reynolds. Scheduling partially ordered tasks with probabilistic execution times. In *SOSP '75: Proceedings of the fifth ACM symposium on Operating systems principles*, pages 169–177, New York, NY, USA, 1975.

[11] Charles E. Clark. The greatest of a finite set of random variables. *Operations Research*, 9(2):145–162, March/April 1961.

[12] Richard Cole and Ramesh Hariharan. Dynamic LCA Queries on Trees. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 235–244. Society for Industrial and Applied Mathematics Philadelphia, PA, USA, 1999.

[13] Josep Díaz, Jordi Petit, and Maria J. Serna. A Survey of Graph Layout Problems. *ACM Computing Surveys*, 34(3):313–356, 2002.

[14] Bajis Dodin. Bounding the project completion time distribution in PERT networks. *Operations Research*, 33(4):862–881, July 1985.

[15] D. R. Fulkerson. Expected Critical Path Lengths in PERT Networks. *Operations Research*, 10(6):808–817, November 1962.

[16] Harold N. Gabow. Data Structures for Weighted Matching and Nearest Common Ancestors with Linking. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443, 1990.

[17] Jane N. Hagstrom. Computational complexity of PERT problems. *Networks*, 18(2):139–147, 1988.

[18] Mark C. Hansen, Hakan Yalcin, and John P. Hayes. Unveiling the iscas-85 benchmarks: A case study in reverse engineering. *IEEE Design & Test*, 16(3):72–80, 1999.

[19] Francis B. Hildebrand. *Introduction to Numerical Analysis.* Dover Publications, 2 edition, 1987.

[20] Jerzy Kamburowski. Normally Distributed Activity Durations in PERT Networks. *The Journal of the Operational Research Society*, 36(11):1051–11057, November 1985.

[21] George B. Kleindorfer. Bounding Distributions for a Stochastic Acyclic Network. *Operations Research*, 19(7):1586–1601, November 1971.

[22] R. Kolisch and A. Sprecher. Psplib - a project scheduling library. *European Journal of Operational Research*, 96:205–216, 1996.

[23] Mukkai S Krishnamoorthy and Narsingh Deo. Complexity of the minimum-dummy-activities problem in a pert network. *Networks*, 9(3):189–194, 1979.

[24] Kenli Li, Xiaoyong Tang, B. Veeravalli, and Keqin Li. Scheduling precedence constrained stochastic tasks on heterogeneous cluster systems. *Computers, IEEE Transactions on*, 64(1):191–204, Jan 2015.

[25] Arfst Ludwig, Rolf H. Möhring, and Frederik Stork. A Computational Study on Bounding the Makespan Distribution in Stochastic Project Networks. *Annals of Operations Research*, 102(1–4):49–64, February 2001.

[26] D. G. Malcolm, J. H. Roseboom, Charles E. Clark, and W. Fazar. Application of a Technique for Research and Development Program Evaluation. *Operations Research*, 7(5):646–669, September 1959.

[27] J. J. Martin. Distribution of the Time through a Directed, Acyclic Network. *Operations Research*, 13(1):46–66, January 1965.

[28] Makoto Matsumoto and Takuji Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[29] Rolf H Möhring. Scheduling under uncertainty: Bounding the makespan distribution. In *Computational Discrete Mathematics*, pages 79–97. Springer, 2001.

[30] Douglas C. Montgomery and George C. Runger. *Applied Statistics and Probability for Engineers.* Wiley, 5 edition, 2010.

[31] Alfred Müller and Dietrich Stoyan. *Comparison Methods for Stochastic Models and Risks.* Wiley, 2002.

[32] Michael L. Pinedo. Stochastic Scheduling with Release Dates and Due Dates. *Operations Research*, 31(3):559–572, 1983.

[33] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Springer Publishing Company, Incorporated, 2008.

[34] Pierre Robillard and Michel Trahan. Expected Completion Time in PERT Networks. *Operations Research*, 24(1):177–182, January 1976.

[35] Sachin S. Sapatnekar and Hongliang Chang. Statistical Timing Analysis Under Spatial Correlations. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1467–1482, September 2005.

[36] Gilbert Saporta. *Probabilités, analyse des données et statistiques*. Editions Technip, 2006.

[37] D. Sculli. The Completion Time of PERT Networks. *The Journal of the Operational Research Society*, 34(2):155–158, February 1983.

[38] M. A. Stephens. Tests based on edf statistics. In R. B. D'Agostino and M. A. Stephens, editors, *Goodness-of-Fit Techniques*, pages 97–193. Marcel Dekker, New York, 1986.

[39] Thomas G. Stockham. High-speed convolution and correlation. In *Proceedings of the AFIPS Sprinp Joint Computer Conference*, pages 229–233, 1966.

[40] Xiaoyong Tang, Kenli Li, Guiping Liao, Kui Fang, and Fan Wu. A stochastic scheduling algorithm for precedence constrained tasks on grid. *Future Generation Computer Systems*, 27(8):1083–1091, 2011.

[41] Takao Tobita and Hironori Kasahara. A standard task graph set for fair evaluation of multiprocessor scheduling algorithms. *Journal of Scheduling*, 5(5):379–394, 2002.

[42] Richard M. van Slyke. Monte Carlo Methods and the PERT Problem. *Operations Research*, 11(5):839–860, September/October 1963.

[43] Chandu Visweswariah, Kaushik Ravindran, Kerim Kalafala, Steven G. Walker, Sambasivan Narayan, Daniel K. Beece, Jeff Piaget, Natesan Venkateswaran, and Jeffrey G. Hemmett. First-order incremental block-based statistical timing analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(10):2170–2180, October 2006.

[44] Richard R. Weber. Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flowtime. *Journal of Applied Probability*, 19(1):167–182, 1982.

[45] Gideon Weiss. Stochastic bounds on distributions of optimal value functions with applications to pert, network flows and reliability. *Annals of Operations Research*, 1(1):59–65, 1984.

[46] Ming-Jong Yao and Weng-Ming Chu. A new approximation algorithm for obtaining the probability distribution function for project completion time. *Computers & Mathematics with Applications*, 54(2):282–295, 2007.

[47] Nihal Yazici-Pekergin and Jean-Marc Vincent. Stochastic Bounds on Execution Times of Parallel Programs. *IEEE Transactions on Software Engineering*, 17(10):1005–1012, October 1991.

[48] Lizheng Zhang, Weijen Chen, Yuhen Hu, and Charlie Chung-ping Chen. Statistical static timing analysis with conditional linear max/min approximation and extended canonical timing model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1183–1191, June 2006.

[49] Wei Zheng and Rizos Sakellariou. Stochastic dag scheduling using a monte carlo approach. *Journal of Parallel and Distributed Computing*, 73(12):1673–1689, 2013.

7 Emapse

In order to evaluate the quality of CorLCA and Cordyn, we have developed a software platform, *Emapse* (Evaluation of MAx-Plus Stochastic Expression) for evaluating the distribution of the longest path length of a directed acyclic graph with random weights. This platform implements several methods of the literature in a generic way and allows an experimental comparison of these methods on given instances. It represents 7 400 lines of Java and is available on https://gforge.inria.fr/projects/emapse/. The related code, data and analysis are available in [8].

7.1 Input/Output

Emapse handles several types of directed acyclic graphs, namely these coming from projects (in the case of project management), these coming from task graph scheduling and these coming from digital circuits design.

Concerning the random dimension, several laws are available for the weights in a graph: uniform, normal, exponential, beta, gamma, Dirac, Weibull and triangular. Their probability density functions and their cumulative distribution functions have all been implemented (except for the beta cumulative distribution function as it requires the incomplete beta function, which involves non-trivial numerical computations).

Once a method estimates the distribution of the longest path length, Emapse can measure several properties on the result (expected value, standard deviation, skewness, kurtosis) and compare it to another random variable using one of the four implemented metrics (which are the error metrics described in Section 8.1).

It is also possible to represent general arithmetic expressions on random variables. Indeed, some methods are not specific to directed acyclic graphs with random weights (such as the Monte-Carlo method or Cordyn, which can be generalized to handle general expressions) and therefore can be used for this kind of input.

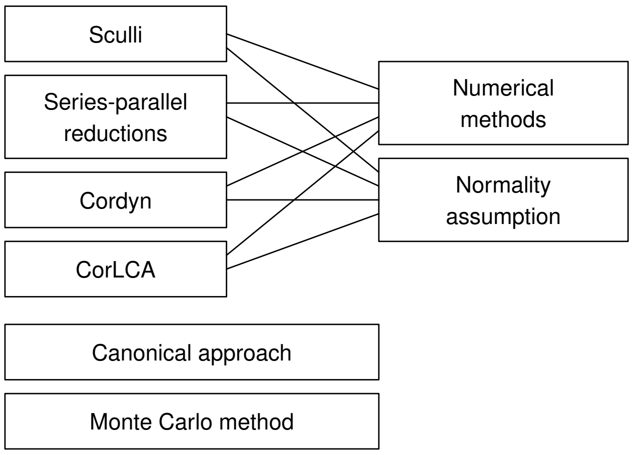


Figure 12: Methods implemented in Emapse. On the left are the different ways to handle dependencies of the intermediate results and on the right are the different methods for evaluating arithmetic expressions.

7.2 Methods

On Figure 12, we have an overview of all the implemented methods. We distinguish the ways dependencies are handled from the methods used to compute arithmetic operation. Indeed, it is possible to assume the normality (Sec. 3.2.2) while still doing series-parallel reductions (Sec. 3.2.1). Conversely, Sculli's approach (Sec. 3.2.2) can be performed using numerical methods (Sec. 3.1.1). However, the mixing of these two types of approaches is not possible for the canonical approach (Sec. 3.2.3) and the Monte Carlo method (Section 3.4).

7.3 Numerical Computations

Numerical computations used in the methods described in Section 3.1.1 have some pitfalls. We recall that a sum requires a convolution. The convolution time complexity is reduced by handling probability density in the frequency domain. When two functions are defined on intervals of different sizes, it is required to interpolate the data before doing the Fourier transform. This can lead to either some loss of precision or an important increase of the number of values (depending on the adopted strategy). In Emapse, we implemented the *OverLap-Add* method [39], which performs a convolution efficiently in a fragmented way when probability densities are defined on intervals of different sizes.

Moreover, we use the Mersenne Twister [28] as a pseudo-random generator for the Monte-Carlo method.

8 Error Metrics

For each described instance, twelve methods are executed among which the Monte Carlo method with one million iterations. The result of this method serves as the reference (the difference between this result and the exact distribution of the longest path length is analyzed in Section 3.4).

8.1 Metric Descriptions

There exist several ways to quantify the error committed by each method by comparing the produced result to the reference result. We study below several errors metrics.

It is possible to derive three types of relative errors: error of the expected value, error of the standard deviation, and error of the skewness. For instance, the relative error of the expected value is noted $\left|\frac{\hat{\mu}-\mu}{\mu}\right|$ where μ is the mean of the values obtained with the Monte Carlo method and $\hat{\mu}$ is the expected value estimated by a given method.

In addition to these three direct metrics, probability and statistic theories provide four other metrics. Let F be the empirical cumulative distribution function (ECDF) obtained with the Monte Carlo method and $\hat{F}$ the cumulative distribution function estimated by a given method.

Anderson-Darling statistic
$A^2 = -T - S$ where $S = \frac{1}{T}\sum_{k=1}^{T}(2k-1) \times (\ln(\hat{F}(X_k)) + \ln(1-\hat{F}(X_{T+1+k})))$, T is the number of iterations of the Monte Carlo method and X_k is the k^{th} element in the list of ordered values of this same method [38]. Its range is $[0,\infty[$. If the supports of both compared random variables are distinct, then the method involves the computation of the logarithm of zero, which leads to a failure.

Kolmogorov-Smirnov statistic
$D = \sup_x |F(x) - \hat{F}(x)|$ [2, Section 7.3.2]. Its range is $[0,1]$.

Cramér-von-Mises statistic
$\omega^2 = \frac{1}{n}\int(F(x) - \hat{F}(x))^2 d\hat{F}(x)$ [1]. Its range is $[0,\frac{1}{3}]$.

Hellinger distance $H = \sqrt{(1-BC)}$
where $BC = \int_x \sqrt{f(x) \times \hat{f}(x)}dx$ is the Bhattacharyya coefficient [4]. This metric requires the probability density functions of both compared random variables. As the Monte Carlo method produces an ECDF, a numerical estimation must be done. Its range is $[0,1]$.

For each of these four metrics, a high value means that both random variables are significantly different, while a value close to zero indicates that they are similar.

8.2 Empirical Comparison

In the previous section, three relative error metrics (expected value, standard deviation, and skewness) and four other probabilistic and statistic metrics (the Hellinger distance and three other statistics) are presented. We compare them in this section.

Figure 13 shows the correlations between these error metrics when we use them to quantify the precision of eleven methods (all possible combinations of Figure 12 in which there are two variations for series-parallel reductions depending on whether Kamburowski's minimization method [3] is used or not) over all the generated instances. On this figure, the seven metrics described in Section 8.1 are compared pairwise. We obtain then a squared matrix of 7×7 plots (due to symmetry, only 21 are represented in the

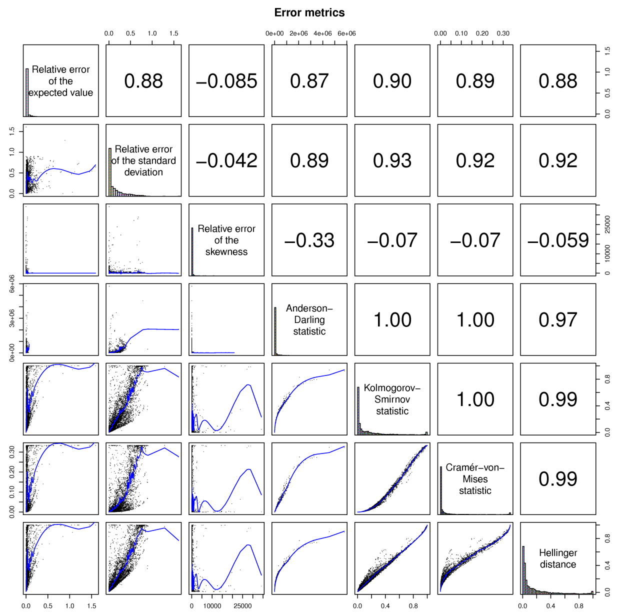


Figure 13: Correlations between error metrics when applied on eleven methods over 1260 instances. Lower-left part: plots with 13 860 measures (with cubic spline interpolation). Upper-right part: Spearman correlation coefficients.

lower-left part). The name and the histogram of the measured values of each metric is given on the diagonal.

The upper-right part of the matrix contains the Spearman correlation coefficients between the metrics specified by the line and the column. The higher the correlation, the closer this coefficient is to one. This correlation coefficient is more robust than Pearson's coefficient when the relation between two metrics is not linear (e.g., between the Kolmogorov-Smirnov statistic and the Cramér-von-Mises statistic).

The case of the expected value relative error is representative. As the measured values are all closed to zero, it is difficult to distinguish the dependence of this metric with the others. However, the high values of its Spearman's coefficients (between 0.88 and 0.90, skewness put apart) suggest that when other metrics have high values, then this one is more likely to have a high value, and reciprocally.

More generally, all measures are (highly) correlated with the exception of the skewness relative error. We postulate that the other six metrics are all relevant for quantifying error, but possibly on different aspects.

We also remark that the four probabilistic and statistic metrics are almost equivalent. Indeed, correlation coefficients between Anderson-Darling statistics, Kolmogorov-Smirnov statistics and Cramér-von-Mises statistics are all equal to 1.00. Only the Hellinger distance differs slightly in this regard.

As these four metrics take into account the complete distribution in their computation, we prefer them to the first three relative error metrics. Among these four metrics, the Kolmogorov-Smirnov statistic and the Hellinger distance present two advantages: their supports have a better definition (i.e., it is $[0, 1]$ in both cases) and the measured values have a better dispersion in this interval (as can be seen on their histograms). Moreover, the Kolmogorov-Smirnov statistic can directly be applied on ECDF and the Kolmogorov distribution allows us to state that the computation of this statistic is precise to a quantity 1.629‰ with one million of iterations (see Section 3.4).

Therefore, the selected error metric of this paper is the Kolmogorov-Smirnov statistic.