Accuracy vs. Complexity: the stochastic bound approach ⋆

F. Ait Salaht * **J. Cohen** ** **H. Castel Taleb** ***
J.M. Fourneau **** **N. Pekergin** †

* *PRiSM, Univ. Versailles St Quentin, UMR CNRS 8144, Versailles France (e-mail: safa@prism.uvsq.fr).*
** *PRiSM, Univ. Versailles St Quentin, UMR CNRS 8144, Versailles France (e-mail: joco@prism.uvsq.fr).*
*** *SAMOVAR, UMR 5157, Télécom Sud Paris, Evry, France, (e-mail: hind.Castel@it-sudparis.eu)*
**** *PRiSM, Univ. Versailles St Quentin, UMR CNRS 8144, Versailles France (e-mail: jmf@prism.uvsq.fr).*
† *LACL, Univ. Paris Est, Créteil, France (e-mail: nihal.pekergin@u-pec.fr).*

Abstract: We present an algorithmic technique based on stochastic ordering to obtain upper and lower bounding distributions for the results of some optimisation problems on discrete random variables which are hard to solve exactly due to the multiplicative increasing size of the distribution at each step. We illustrate the approach with the distribution of the completion time of a task graph.

Keywords: Performance analysis, Bounding method, Probability distribution function, Stochastic approximation, Numerical analysis, Algorithms, Graph theoretic models.

1. INTRODUCTION

We want to develop an algorithmic approach to obtain stochastic bounds rather than approximations for some problems in Operation Research which deal with random variables and which are computationally hard to solve numerically. To be more precise, we consider random variables with discrete distributions. Many optimisation problems on these random variables make the size of the distribution increases multiplicatively at each iteration. Intuitively this is the reason of the hardness (and sometimes the NP-hard property). We propose to use the stochastic bound theory to reduce the size of the distributions at each step of the computation. The control on the size of the distribution of the results will help to control the complexity while the stochastic bounds theory insures that the result is a bound of the exact distribution. We illustrate the approach by a well-known problem in performance evaluation: the distribution of the completion time in a stochastic task graph. It must be clear that we are interested in the presentation of the methodology and its application to a well-known example rather than an extensive comparison of results on the computation of bounds and approximate results for this particular problem. We refer to O'Connor (2006) and Baccelli et al. (1993) for this topic.

More precisely, let $\boldsymbol{d}$ be the probability mass function (*pmf* in the following) for a discrete distribution of probability on a discrete state space totally ordered $\mathcal{H}$. Assume that the cardinal of $\mathcal{H}$ is N. We also assume that the distribution is such that $\boldsymbol{d}(i) > 0$ for $i \in \mathcal{H}$: no element of $\mathcal{H}$ has a zero probability. $\boldsymbol{d}$ is considered as a vector of size N whose entries are positive. In general, set $\mathcal{H}$ consists in an interval of integers. Let $\boldsymbol{r}$ be a positive increasing reward. We want to compute the *pmf* of a distribution on a support (say $\mathcal{F}$) with K states, such that $K < N$, which is the best approximation of $\boldsymbol{d}$ for $\boldsymbol{r}$ and which is a stochastic lower bound. Of course, we also solve the problem for an upper bound but we present most of the results, here, for a lower bound, for the sake of readability. Let $F_{\boldsymbol{d}}$ denote the cumulative distribution function for an arbitrary distribution $\boldsymbol{d}$. Let as denote $R_{\boldsymbol{d}} = \sum \boldsymbol{r}(i).\boldsymbol{d}(i)$. Finally, let $\mathcal{G} = \mathcal{H} \cup \mathcal{F}$.

We want to find the distribution $\boldsymbol{db}$ on states of $\mathcal{F}$ which is a stochastic lower bound of the exact distribution $\boldsymbol{d}$ and such that $R_{\boldsymbol{db}}$ is the most accurate, when we compute the expected reward. As the distribution $\boldsymbol{db}$ is on a smaller state space, the operations on the bounding distributions will be easier to compute. Since we compute stochastic bounds, we obtain a bound on the results if the operations involved in the problem are monotone. For instance the computation of the distribution for the completion time of a stochastic task graph require two operations: the addition and the maximum of the random variables, which are respectively associated with the convolution and the product of underlying pmf when random variables are mutually independent. Both operations are monotone. Thus, stochastic bounds after numerical computations at each iteration provide a stochastic bound of the final result. The reward $\boldsymbol{r}$ is the criterion to optimise to choose the bounding distribution. The reward is supposed to

⋆ This work is partially supported by a grant from DIGITEO, project MARINA 2010.

be positive and increasing because such properties are consistent with the stochastic ordering of distributions as we will show in Section 2. Many rewards used in performance modelling are positive and increasing: for instance the expectation and the other moments.

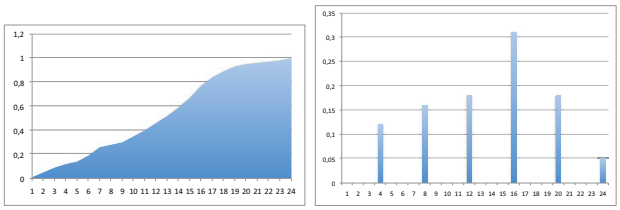

Fig. 1. Discretization of a Continuous Distribution (left) and Representation as a PH distribution (right).

A similar approach has been recently proposed in (Tancrez et al. (2009)) with a slightly different context and approach. Let us first introduce their problem. They begin with a continuous distribution which models the service duration in a production line. They build an upper bounding discrete distribution as follows. They divide the support into K equal subintervals. Each of these subintervals of the continuous distribution is associated with one single point of the discrete one. This point is the upper limit of the interval and the probability mass of the sub-interval is associated to that point (see Fig. 1). Such a mapping of the points and the mass provides an upper bound for the st-ordering. Once the discrete distribution is built, a PH distribution is easily associated to it and a numerical analysis of lines of queues is conducted. As the production lines considered can be modeled as a decision free stochastic Petri-nets, it is known, since the seminal work of Baccelli et al. (1992) that bounding the distribution of the service times in the queues provides a bound on the end to end delay. To conclude with their approach, their bound is not optimal, but it is the most regular one in some sense as it is based on equal subintervals and the problem is to compute K integrals of the distribution on the K sub-intervals. Our approach differs as we consider discrete distribution as inputs and we want to compute the most accurate approximation at each level according to a reward function.

The technical part of the paper is organised as follows. In the next section we introduce the strong stochastic ordering among random variables and distributions. Section 3 is devoted to the computation of the most accurate bound according to the expectation of a reward. We first define the problem and give some theoretical results on the bounds. Then, we propose a greedy algorithm and we prove that, under some conditions, the algorithm is optimal. We also show that if the conditions do not hold, the solution is far from the best one. Finally we present a dynamic programming approach which always gives the optimal solution but whose complexity is cubic. In Section 4 we show how to use this approach to compute upper and lower stochastic bounds on the distribution of the completion time of a serie-parallel task graph.

2. INTRODUCTION TO STOCHASTIC BOUNDS

We present briefly the stochastic comparison method and we refer to the book (Muller and Stoyan (2002)) for the theoretical issues and several applications of this method. We consider that state space $\mathcal{G}$ is endowed with a total order denoted as $\leq$. Let X and Y be two discrete random variables taking values on $\mathcal{G}$, with cumulative probability distributions F_X and F_Y, and probability distributions $\boldsymbol{p}_X$ and $\boldsymbol{p}_Y$ ($\boldsymbol{p}_X(i) = Prob(X = i)$, and $\boldsymbol{p}_Y(i) = Prob(Y = i)$, for $i \in \mathcal{G}$). The strong stochastic ordering $\leq_{st}$ is defined as follows :

Definition 1. $X \leq_{st} Y \iff \mathbb{E}f(X) \leq \mathbb{E}f(Y)$ for all non decreasing functions $f : \mathcal{G} \to \mathbb{R}^+$ whenever expectations exist.

The strong stochastic ordering $\leq_{st}$ can be also defined from the distribution functions as follows :

Definition 2. $X \leq_{st} Y \Leftrightarrow F_X(a) \geq F_Y(a),\ \forall a \in \mathcal{G}$

We suppose now that $\mathcal{G} = \{1, 2, \ldots, n\}$.

Proposition 3. The stochastic ordering between probability measures $\boldsymbol{p}_X$ and $\boldsymbol{p}_Y$ is defined as follows:

$$\boldsymbol{p}_X \leq_{st} \boldsymbol{p}_Y \Leftrightarrow \forall i,\ 1 \leq i \leq n,\ \sum_{k=i}^{n} \boldsymbol{p}_X(k) \leq \sum_{k=i}^{n} \boldsymbol{p}_Y(k) \quad (1)$$

Example 1. See Fig. 2. We consider $\mathcal{G} = \{1, 2, \ldots, 7\}$, and two discrete probability distributions $\boldsymbol{p}_X = [0.1,\ 0.2,\ 0.1,\ 0.2,\ 0.05,\ 0.1,\ 0.25]$, and $\boldsymbol{p}_Y = [0,\ 0.25,\ 0.05,\ 0.1,\ 0.15,\ 0.15,\ 0.3]$. We can easily verify that $\boldsymbol{p}_X \leq_{st} \boldsymbol{p}_Y$: the probability mass of $\boldsymbol{p}_Y$ is concentrated to higher states such as the cumulative distribution of $\boldsymbol{p}_Y$ is always below the cumulative distribution of $\boldsymbol{p}_X$.

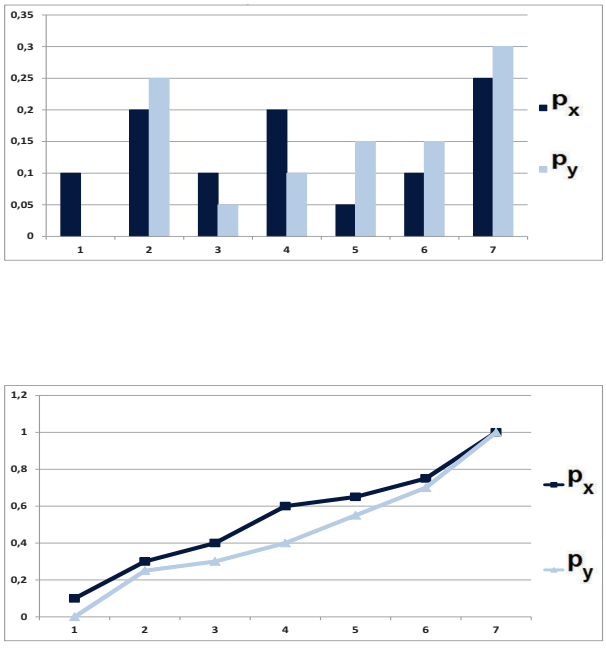

Fig. 2. $\boldsymbol{p}_X \leq_{st} \boldsymbol{p}_Y$: Their pmf (up) and their cumulative distribution functions (down) .

3. MOST ACCURATE STOCHASTIC BOUNDS

We first present some theoretical results then we prove some algorithms. In the following we will prove the results

for the lower bound. The results for the upper bound are given without proofs for the sake of readability.

3.1 Theoretical results

In the following we will characterize the distributions $\boldsymbol{d}1$ and $\boldsymbol{d}2$ which minimize the difference of the expected rewards. More formally, we want to answer the following question: given a distribution $\boldsymbol{d}$ having state-space $\mathcal{H}$, compute $\boldsymbol{d}1$ and $\boldsymbol{d}2$ such that:

(1) $\boldsymbol{d}2 \leq_{st} \boldsymbol{d} \leq_{st} \boldsymbol{d}1$,
(2) the supports of $\boldsymbol{d}1$ and $\boldsymbol{d}2$ have only K states (not the same set for both distributions, but the same size),
(3) $\sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}2(i)$ is minimal among the set of distributions on K states that are stochastic lower bound of $\boldsymbol{d}$,
(4) $\sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}1(i) - \sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}(i)$ is minimal among the set of distributions on K states that are stochastic upper bound of $\boldsymbol{d}$. Note that we do not know, at this step, that the set $\mathcal{G}$ is the same for upper and lower bounds.

$\boldsymbol{d}1$ and $\boldsymbol{d}2$ will be denoted as the optimal bounding distributions on K states according to reward $\boldsymbol{r}$.

Proposition 4. Note that as $\boldsymbol{r}$ is increasing and $\boldsymbol{d}2 \leq_{st} \boldsymbol{d}$ we know that $\sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}2(i)$ is positive.

Remember that $\mathcal{G} = \mathcal{H} \cup \mathcal{F}$. As the states of $\mathcal{G}$ are totally ordered and finite, we have a minimal and a maximal state. They are respectively denoted as *MinState* and *MaxState*.

Proposition 5. Suppose that we compute the more accurate lower bound, then state *MinState* is in $\mathcal{F} \cap \mathcal{H}$: i.e. $\boldsymbol{d}2(\mathit{MinState}) > 0$ and $\boldsymbol{d}(\mathit{MinState}) > 0$. Furthermore state *MaxState* is in $\mathcal{H}$.

Proof. Remember that *MinState* is defined as the minimal state of $\mathcal{F}\cup\mathcal{H}$. Thus we must prove that $\boldsymbol{d}2(\mathit{MinState}) \neq 0$ and $\boldsymbol{d}(\mathit{MinState}) \neq 0$ to make sure that *MinState* is in $\mathcal{F} \cap \mathcal{H}$. As $\boldsymbol{d}2 \leq_{st} \boldsymbol{d}$, we get $\boldsymbol{d}2(\mathit{MinState}) \geq \boldsymbol{d}(\mathit{MinState})$.

- If $\boldsymbol{d}(\mathit{MinState}) > 0$, then $\boldsymbol{d}2(\mathit{MinState}) > 0$. Thus the state is both in $\mathcal{H}$ and $\mathcal{F}$.
- If if $\boldsymbol{d}(\mathit{MinState}) = 0$, then we have $\boldsymbol{d}2(\mathit{MinState}) > 0$ as state *MinState* is in $\mathcal{G}$. There is a contradiction with the optimality of $\boldsymbol{d}2$, as we can easily build a bound better than $\boldsymbol{d}2$.

Now assume that *MaxState* is not in $\mathcal{H}$. Remember that we have $\boldsymbol{d}2(\mathit{MaxState}) \leq \boldsymbol{d}(\mathit{MaxState})$ due to the stochastic ordering. Therefore if $\boldsymbol{d}(\mathit{MaxState}) = 0$, then $\boldsymbol{d}2(\mathit{MaxState}) = 0$ and state *MaxState* is neither in $\mathcal{F}$ nor in $\mathcal{H}$. Thus *MaxState* is not a state of $\mathcal{G}$. This is a contradiction. □

We obtain similar results for distribution $\boldsymbol{d}1$ with an inversion of the states *MinState* and *MaxState*. Let us now prove the most important result about the sets. First define the predecessor and successor functions. Let x be an arbitrary state. We define the predecessor k of x in $\mathcal{G}$ as the biggest state in $\mathcal{G}$ smaller than x: $k = \Gamma^-{}_{\mathcal{G}}(x)$. Similarly, the successor of x is the smallest state of $\mathcal{G}$ bigger than x: $\Gamma^+{}_{\mathcal{G}}(x)$.

Lemma 6. Let $\boldsymbol{d}2$ be the optimal distribution solution. Assume that the support of $\boldsymbol{d}$ is $\mathcal{H}$ and the support of $\boldsymbol{d}2$ is $\mathcal{F}$, then $\mathcal{F} \subset \mathcal{H}$ (i.e. $\mathcal{G} = \mathcal{H}$).

Proof. By contradiction, assume that $\mathcal{G} \neq \mathcal{H}$. Let x be the smallest element in $\mathcal{F}$ which is not in $\mathcal{H}$. Thus, $\boldsymbol{d}2(x) > 0$ and $\boldsymbol{d}(x) = 0$. Let $y = \Gamma^-{}_{\mathcal{G}}(x)$ and $h = \Gamma^+{}_{\mathcal{G}}(x)$. By construction, y is in $\mathcal{H}\cap\mathcal{F}$. Proposition 5 states that y and h exist in $\mathcal{G}$. We will prove a contradiction by designing a new distribution $\boldsymbol{d}3$ such that $\boldsymbol{d}2 \leq_{st} \boldsymbol{d}3 \leq_{st} \boldsymbol{d}$ and the reward is better for $\boldsymbol{d}3$ than for $\boldsymbol{d}2$. We must consider two cases, according to the fact that h is in $\mathcal{H}$ or not.

- case [$x \in \mathcal{F}$:] We define $\boldsymbol{d}3$ as follows:
$$\left[\begin{array}{lcll} \boldsymbol{d}3(i) & = & \boldsymbol{d}2(i) & \forall i \leq y, \\ \boldsymbol{d}3(x) & = & \boldsymbol{d}2(x)/2 & , \\ \boldsymbol{d}3(h) & = & \boldsymbol{d}2(h) + \boldsymbol{d}2(x)/2 & , \\ \boldsymbol{d}3(j) & = & \boldsymbol{d}2(j) & \forall j \geq h. \end{array}\right.$$
Note that $\boldsymbol{d}3$ uses the same set $\mathcal{F}$ as $\boldsymbol{d}2$. This case is depicted Fig. 3.
- case [$x \notin \mathcal{F}$:] We define $\boldsymbol{d}3$ as:
$$\left[\begin{array}{lcll} \boldsymbol{d}3(i) & = & \boldsymbol{d}2(i) & \forall i \leq y, \\ \boldsymbol{d}3(x) & = & 0 & , \\ \boldsymbol{d}3(h) & = & \boldsymbol{d}2(x) & , \\ \boldsymbol{d}3(j) & = & \boldsymbol{d}2(j) & \forall j \geq h. \end{array}\right.$$
Note that $\boldsymbol{d}3$ does not use the same set as $\boldsymbol{d}2$. But both sets have the same size.

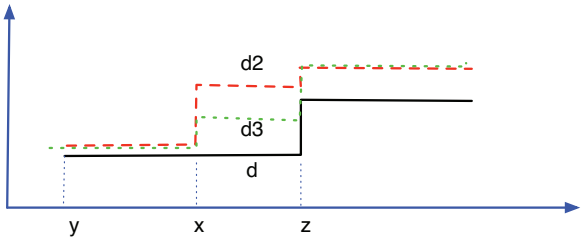

Fig. 3. The cumulative distributions in the first case.

Now it is clear that the support of $\boldsymbol{d}3$ has K states, and a careful inspection of the inequalities shows that $\boldsymbol{d}2 \leq_{st} \boldsymbol{d}3 \leq_{st} \boldsymbol{d}$ and $\boldsymbol{d}3 \neq \boldsymbol{d}2$. Therefore, for an increasing positive reward, we get $R_{\boldsymbol{d}3} > R_{\boldsymbol{d}2}$. We obtain a contradiction: $\boldsymbol{d}2$ is not the optimal bound. Therefore, such a node x does not exist and $\mathcal{F} \subset \mathcal{H}$. □

Lemma 7. Let $\boldsymbol{d}2$ be the most accurate lower bound of $\boldsymbol{d}$ on $N-1$ states for a positive reward $\boldsymbol{r}$. Then, distributions $\boldsymbol{d}$ and $\boldsymbol{d}2$ have the same values on $N-2$ states of $\mathcal{H}$.

Proof. $\boldsymbol{d}2$ is obtained by the minimization of the difference of the expected rewards $\sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i\in\mathcal{G}} \boldsymbol{r}(i)\boldsymbol{d}2(i)$. We know that the supports of $\boldsymbol{d}2$ and $\boldsymbol{d}$ only differ by a state (say k) which is removed from $\mathcal{H}$. Let $l = \Gamma^-{}_{\mathcal{H}}(k)$. l exists because Proposition 5 states that *MinState* is both in $\mathcal{H}$ and $\mathcal{F}$. Therefore $k \neq \mathit{MinState}$ and $\Gamma^-{}_{\mathcal{H}}(k)$ exists.

By the optimality argument that we have previously used in the proof of the previous Lemma, one can easily show that:
$$\left[\begin{array}{lcll} \boldsymbol{d}2(i) & = & \boldsymbol{d}(i) & \forall i \leq l, \\ \boldsymbol{d}2(l) & = & \boldsymbol{d}(l) + \boldsymbol{d}(k) & , \\ \boldsymbol{d}2(j) & = & \boldsymbol{d}(j) & \forall j \geq k. \end{array}\right.$$
The difference between the expected rewards is just $\boldsymbol{d}(k)(\boldsymbol{r}(k) - \boldsymbol{r}(l))$. Thus the method, in that case, consists in computing the values of $\boldsymbol{d}(i)(\boldsymbol{r}(i) - \boldsymbol{r}(\Gamma^-{}_{\mathcal{H}}(i)))$ for all states i in $\mathcal{H}$ and to remove the point which minimizes this difference of cumulated rewards. □

This result is the intuitive explanation of the greedy Algorithm in the next sub-section.

Lemma 8. Removing two adjacent nodes costs more for the cumulated rewards than the sum of the two independent deletions.

Proof. Suppose that we remove nodes x and y from $\mathcal{H}$ to build $\boldsymbol{d}2$.

- If x and y are adjacent, we assume without loss of generality that $x = \Gamma^-{}_{\mathcal{H}}(y)$. Let v be $\Gamma^-{}_{\mathcal{H}}(x)$. The cumulated expected reward becomes: $R_{\boldsymbol{d}} - \boldsymbol{d}(x)(\boldsymbol{r}(x) - \boldsymbol{r}(v)) - \boldsymbol{d}(y)(\boldsymbol{r}(y) - \boldsymbol{r}(v))$.
- Removing x alone subtracts $\boldsymbol{d}(x)(\boldsymbol{r}(x) - \boldsymbol{r}(v))$ out of the reward. And the deletion of y alone subtracts $\boldsymbol{d}(y)(\boldsymbol{r}(y) - \boldsymbol{r}(x))$ of the reward.

Then, comparing the results, one can see that the deletion of adjacent nodes has an extra cost of $\boldsymbol{d}(y)(\boldsymbol{r}(x) - \boldsymbol{r}(v))$. □

3.2 A Greedy Algorithm

We sort the values of the difference of reward (i.e. $\boldsymbol{d}(i)(\boldsymbol{r}(i) - \boldsymbol{r}(\Gamma^-{}_{\mathcal{H}}(i)))$) and we select the $N - K$ first values. We use an array of boolean to keep in memory that a node has been modified during the algorithm. At each step, two nodes are modified: the node which is removed and the node which receives the probability coming from the nodes we remove and which is just before it when we compute a lower bounding distribution.

Algorithm 1 Greedy (sometimes optimal) Lower Bound

1: Let $\boldsymbol{d}2 = \boldsymbol{d}$ and $\boldsymbol{z}(\mathit{MinState}) = \infty$.
2: For all i in $\mathcal{H}$ except *MinState*, compute the value of $\boldsymbol{d}(i)(\boldsymbol{r}(i) - \boldsymbol{r}(\Gamma^-{}_{\mathcal{H}}(i)))$ and store it in vector $\boldsymbol{z}$.
3: Let $\mathcal{F} = \mathcal{H}$, and set OptimalSelection to True.
4: Sort vector $\boldsymbol{z}$ in increasing order.
5: Initialize a vector ***Mark*** indexed by $\mathcal{F}$ to False.
6: **for** $l = 1$ to $N - K$ **do**
7: Let j be the state of index l in $\boldsymbol{z}$.
8: Let $k = \Gamma^-{}_{\mathcal{H}}(j)$ and $m = \Gamma^-{}_{\mathcal{F}}(j)$.
9: **if** ***Mark***(j) or ***Mark***(k) **then**
10: OptimalSelection=False
11: **end if**
12: Let ***Mark***(j)=True; ***Mark***(k)=True.
13: Let $\boldsymbol{d}2(m) = \boldsymbol{d}2(m) + \boldsymbol{d}2(j)$.
14: Remove state j from $\mathcal{F}$. $\boldsymbol{d}2(j)$ is not defined anymore.
15: **end for**

Theorem 9. The greedy algorithm provides a distribution with support $\mathcal{F}$ and values $\boldsymbol{d}2$, which is a strong stochastic lower bound of $\boldsymbol{d}$. The distribution has K points and it is optimal if the value of variable "OptimalSelection" at the end of the algorithm is True. Moreover, the complexity of the greedy algorithm is $O(NlogN)$ due to the Sort operation.

Proof. The first part of the theorem comes from instructions 8 and 13, where we move some probability to a smaller state. This operation makes the distribution a lower bound according to the st-ordering. The assertion about the size is trivial as we remove $N - K$ states out of N. Let us now prove the optimality when the condition holds at the end of algorithm.

Assume that boolean OptimalSelection is true at the end of the algorithm. This implies that we do not remove adjacent nodes in $\mathcal{H}$. Remember that $\boldsymbol{z}$ is sorted in an increasing order. Thus, $R_{\boldsymbol{d}2} = R_{\boldsymbol{d}} - \sum_{l=1}^{N-K} \boldsymbol{z}(l)$. Indeed, if OptimalSelection is true, then, at each step of the algorithm, we have $m = k$ and the modification of $\boldsymbol{d}2$ performed by instruction 13 implies that we subtract $\boldsymbol{d}2(j)(\boldsymbol{r}(j) - \boldsymbol{r}(m))$ out of the cumulative reward $R_{\boldsymbol{d}2}$. Finally, we remark that according to Lemma 8, removing any subset with adjacent nodes will have a higher impact on the expected cumulated reward. Therefore, the solution is optimal. □

Unfortunately, the greedy algorithm does not give the most accurate bound when the optimality criterion in the algorithm is not true as shown in Example 2.

Example 2. Let $\mathcal{H} = \{1, 2, 3, 4, 5, 6\}$, $\boldsymbol{r}(i) = i$, and $\boldsymbol{d} = [0.3, 0.1, 0.1, 0.1, 0.2, 0.2]$. The initial accumulated reward is $R_{\boldsymbol{d}} = 3.4$. We want to compute the more accurate approximation with a distribution on three states. Vector $\boldsymbol{z}$ is equal to [0.3, 0.1, 0.1, 0.1, 0.2, 0.2]. Algorithm Greedy removes states 2, 3 and 4. And we obtain $\mathcal{F} = \{1, 5, 6\}$, $\boldsymbol{d}2a = [0.6, 0.2, 0.2]$, and $R_{\boldsymbol{d}2a} = 2.8$. However, one can easily found that the solution $\mathcal{F} = \{1, 4, 5\}$, and $\boldsymbol{d}2b = [0.5, 0.1, 0.4]$ is a better bound as it is a lower bound of $\boldsymbol{d}$ and the cumulated reward is equal to 2.9. Thus the solution provided by the greedy algorithm is not optimal.

3.3 Optimal Algorithm based on Dynamic Programming

We will transform our problem dealing with a discrete distribution into a graph theory problem. First, we consider the weighted graph $G = (V, E)$ such that:

- V is the set of vertices such that $V = \mathcal{H} \cup \{\mathit{EndState}\}$ where *EndState* is a new state larger than all the states in $\mathcal{H}$.
- E is the set of arcs such that $(u, v) \in E$ if and only if $u < v$ or if $v = \mathit{EndState}$ and $u \in \mathcal{H}$. The weight of arc $e = (u, v)$, denoted by $\boldsymbol{w}(e)$, is defined as follows:

$$\boldsymbol{w}(e) = \begin{cases} \sum_{j \in \mathcal{H}: u < j < v} \boldsymbol{d}(j)(\boldsymbol{r}(j) - \boldsymbol{r}(u)) & \text{if } v \in \mathcal{H}, \\ \sum_{j \in \mathcal{H}: u < j} \boldsymbol{d}(j)(\boldsymbol{r}(j) - \boldsymbol{r}(u)) & \text{otherwise.} \end{cases}$$

For the remaining of this section, we focus on certain paths from state *MinState* to state *EndState* in graph G.

Lemma 10. Let $\boldsymbol{d}_P$ be a distribution on a state set $\mathcal{F}$ such that $\boldsymbol{d}_P \preceq \boldsymbol{d}$. The path P from state *MinState* to state *EndState* through all elements of $\mathcal{F}$ has weight: $\sum_{i \in \mathcal{H}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i \in \mathcal{F}} \boldsymbol{r}(i)\boldsymbol{d}(i)$.

Proof. Let $\mathcal{F} = \{f_0 = \mathit{MinState}, f_1, \ldots, f_{|\mathcal{F}|}\}$. From its definition, path P contains arcs $(\mathit{MinState}, f_1)$, $(f_{|\mathcal{F}|}, \mathit{EndState})$, (f_i, f_{i+1}), for $i = 1 \ldots, |\mathcal{F}| - 1$. The weight of P is the sum of the weights of all its arcs. So it equals to:

$$\sum_{i=0,\ldots,|\mathcal{F}|-1} \sum_{j \in \mathcal{H}: f_i < j < f_{i+1}} \boldsymbol{d}(j)(\boldsymbol{r}(j) - \boldsymbol{r}(f_i)) + \sum_{j \in \mathcal{H}: f_{|\mathcal{F}|} < j \leq f_{\mathbf{MaxState}}} \boldsymbol{d}(j)(\boldsymbol{r}(j) - \boldsymbol{r}(f_i))$$

After some elementary computations, the previous sum is equal to: $\sum_{i\in\mathcal{H}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i\in\mathcal{F}} \boldsymbol{r}(i)\boldsymbol{d}(i)$.

Moreover, from the previous lemma, it is easy to notice that any path P from state *MinState* to state *EndState* corresponds to a distribution $\boldsymbol{d}_P$ as stated now:

Lemma 11. Let P the path from state *MinState* to state *EndState* through all elements of $\mathcal{F}$ in G. There exists a distribution $\boldsymbol{d}_P$ on a state set $\mathcal{F}$ such that $\boldsymbol{d}_P \preceq \boldsymbol{d}$ and the weight of P is $\sum_{i\in\mathcal{H}} \boldsymbol{r}(i)\boldsymbol{d}(i) - \sum_{i\in\mathcal{F}} \boldsymbol{r}(i)\boldsymbol{d}(i)$.

In fact, computing $\boldsymbol{d}2$ is equivalent to compute a shortest path in G from state *MinState* to state *EndState* with K arcs. Guérin and Orda (2002) give such an algorithm based on dynamic programming. Its complexity is $O(N^2K)$.

Theorem 12. The algorithm in Guérin and Orda (2002) provides the optimal solution and its complexity is cubic when K has the same order as N.

4. THE DISTRIBUTION OF THE COMPLETION TIME IN A STOCHASTIC TASK GRAPH

We model a parallel program as an acyclic and directed graph, called a *task graph* (see Pekergin and Vincent (1991)). Nodes of the task graph represent the tasks, and task durations are positive, random variables $S_i, 1 \leq i \leq n$ which are mutually independent. Arcs define the precedence (synchronization) constraints between tasks: there exists an arc from task i to task j if task i must precede task j. A task i can start its execution when all its immediate predecessors terminate their executions. We assume that node 1 (source node) has no predecessor and node n (sink node) has no successor. Let T_i be the completion date for task i and Γ_i^- is the set of tasks which are immediate predecessors of task i.

$$T_i = \max_{j\in\Gamma_i^-}\{T_j\} + S_i \tag{2}$$

Since the graph is acyclic, task completion times can be derived consecutively and T_n is the overall execution time of the graph. We apply $(max, +)$ operations on random variables. The main difficulty comes from the max operator applied on random variables which are not independent. Even if the individual task execution times S_i are independent, if two paths use a common task j, they are not independent. The computation of the task graph execution time distribution with an infinite (sufficiently large) number of processors is equivalent to the computation of the completion time distribution of a stochastic PERT network which is in the class of $\#P$-complete problems (see O'Connor (2006)). The exact solution can be obtained only by exhaustive enumeration. Hence the exact analysis of such networks is only possible when the network size is small or when the network has a constrained topology. Here we consider serie-parallel task graphs. For such a model, the task completion times can be derived by applying $(max, +)$ operations on independent random variables. They also represent many interesting problems by their own, and they are the basic objects used in many bounding methods for general graphs. Let us explain following O'Connor (2006), how one computes a task graph completion time when task execution times S_i are discrete random variables and the network is an arbitrary DAG. The first step consists in determining the subset of tasks called conditioning set $\mathcal{C}$, that eliminates the dependencies in max operations. After conditioning on the duration of tasks in $\mathcal{C}$, the task graph becomes a series-parallel graph. Clearly all the task execution time configurations for $\mathcal{C}$ tasks must be enumerated for the conditioning, and for each configuration, conditional task graph execution time must be computed. This is the reason that we cannot have polynomial algorithms for the general case.

Although for series-parallel graphs, $\mathcal{C}$ is empty, their computation are not that simple. Indeed we have a linear number of operations "max" and "+" and both operations have a polynomial complexity on the size of the distributions. However the size of the distribution grows rapidly after each operation. Let X (resp. Y) be a discrete random variable taking values in a set $\mathcal{G}_X$ (resp. $\mathcal{G}_Y$) of size $l_X > 1$ (resp. $l_Y > 1$). The cumulative distribution will be noted by $F_X(a) = Prob(X \leq a)$.

Proposition 13. The computation of the distribution of the maximum of two independent random variables requires $O(l_X + l_Y)$ operations (max) and at most $l_X + l_Y - 1$ states for the resulting distribution. The convolution of two independent random variables requires $O(l_X \times l_Y)$ operations $(+)$ and at most $l_X \times l_Y$ states for the resulting distribution.

Proof. The first property comes from a well-kown property: for all $a \in \mathcal{G}_X \cup \mathcal{G}_Y$, $F_{max\{X,Y\}}(a) = F_X(a).F_Y(a)$. Moreover, the addition of two random variables is associated to the convolution of distributions:

$$F_{X+Y}(a) = \sum_{b\in\mathcal{G}_X} Prob(X = b). \sum_{c\in\mathcal{G}_Y | c+b=a} Prob(Y = c)$$

.

Therefore, even for series-parallel graphs, the computation complexity for a detailed distribution on large supports of the overall task execution time may be very expensive. For instance, for a series-parallel graph with m tasks in each branch, if the execution time for each task is a discrete distribution of size p, then the distribution for the execution time of a branch may be in the worst case of size p^m. Thus the distribution size may increase exponentially with the number of nodes.

We propose to apply the bounding algorithms in the previous sections to control the size of the support of the distribution. Thus, after each $(max, +)$ operation we apply either the greedy algorithm given in sub-section 3.2 or the optimal algorithm given in sub-section 3.3 to limit the resulting distribution size to a given threshold K. We must first show that we can replace input parameters of $(max, +)$ operations by bounding distributions on smaller support sizes to compute bounds on the final results. Task graphs constitutes a special case of stochastic timed event graphs that have been largely studied in chapter 8 of Baccelli et al. (1992). Due to (max, +) operations, we have the following monotonicity result, which justifies that we obtain bounds of the distribution of the global execution times using stochastic bounds after each operation.

Property 1. In equation 2, if $\forall j \in \Gamma_i^-, \bar{T}_j \leq_{st} T_j$ and $\bar{S}_i \leq_{st} S_i$ then $\bar{T}_i = \max_{j\in\Gamma_i^-}\{\bar{T}_j\} + \bar{S}_i \leq_{st} T_i$.

We consider a series-parallel task graph composed of b branches with m tasks in each branch. Task duration time

is a discrete random variable of size 4 and these values are chosen randomly as a real value in the interval $[0, 10[$. The overall task graph execution time can be computed by applying $(max, +)$ operations on independent random variables as follows : $S_1 + \max_{1 \leq j \leq b}\{L_j\} + S_{j.m+2}$, where the execution time of the jth branch is $L_j = \sum_{i=(j-1)m+2}^{jm+1} S_i$.

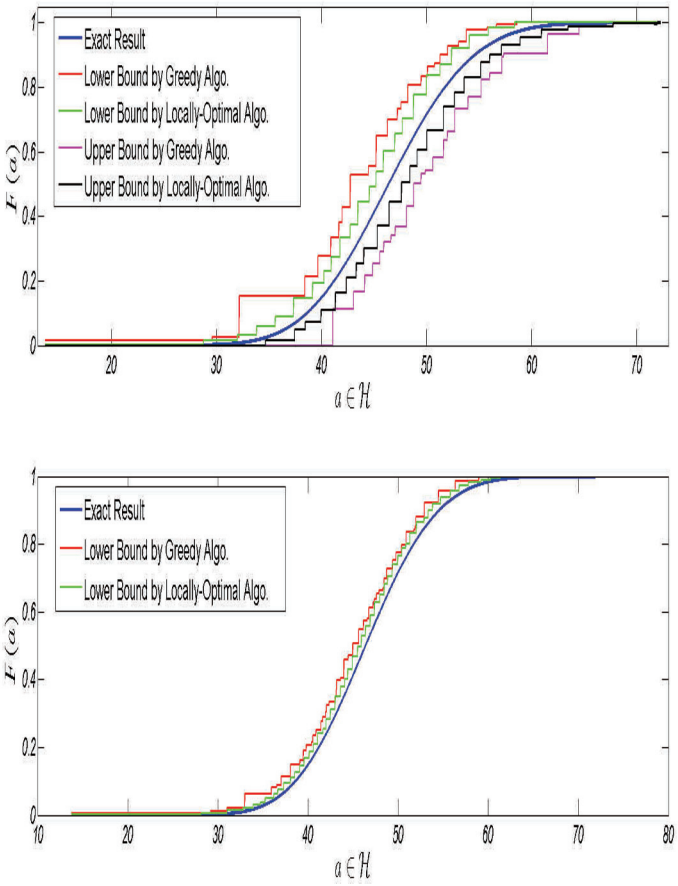

Fig. 4. Upper and Lower bounds of the cumulative distributions for m=7 and K=25 (up), Lower bounds for m=7 and K=50.

In Table 1, we give the exact results for task graphs with $b = 3$ branches and various m. Column L is the length of the final distribution, column T reports the computation time in seconds, and column $R_{\boldsymbol{d}}$ contains the reward we have computed: the expected value of the overall task execution time. Table 2 contains the corresponding lower

m	L	T	$R_{\boldsymbol{d}}$
4	12160	0.7383	37.1455
5	46256	7.8542	43.3317
6	188416	415.1603	46.3308
7	785504	$8.3653\ 10^3$	46.5201
8	2974896	$2.4244\ 10^5$	56.1796

Table 1. Exact results

bounding values with $K = 25$ and $K = 50$. We present the results of the greedy algorithm and the dynamic programming algorithm. Clearly, the bounding algorithms are much faster than the exact one. The computation time with greedy bounds does not increase significantly with the size of the graph while it increases with the locally

m	K	Greedy		(Locally)-Optimal	
		T	$R_{\boldsymbol{d2}}$	T	$R_{\boldsymbol{d2}}$
4	25	0.1125	35.6090	0.5781	36.3648
	50	0.1705	36.5403	3.7996	36.8294
5	25	0.1412	41.4151	0.8191	42.2156
	50	0.2484	42.5091	6.0513	42.8496
6	25	0.1793	43.6972	1.0872	45.0021
	50	0.3083	45.0599	8.1150	45.7225
7	25	0.2134	42.9925	1.3683	44.7492
	50	0.3697	45.0117	10.1021	45.7387
8	25	0.2552	52.2219	1.4004	53.9880
	50	10.5801	54.1708	13.4566	55.0026
	100	1.1274	55.0394	94.0466	55.5080

Table 2. Bounds

optimal algorithm but it remains largely smaller than the exact computation time. The accuracy of the bounds are improved for larger values of K, and for $K = 50$ the quality of bounds (greedy and locally optimal) are comparable and they are both accurate. We want to emphasize here, that the dynamic programming algorithm provides an optimal bound for each intermediate result, and we do not have any proof that using several times this method is sufficient to obtain the best bound on K states of the exact distribution. It is the reason why we denote this method as a locally optimal bound.

5. CONCLUSION

Our method has a strong impact on the computation times since we can control distribution sizes. Moreover we can make a trade-off between accuracy and speed by changing distribution sizes. We will develop now new applications in networks performance evaluation based on discretized real measurements such as the approach developed in Hernández-Orallo and Vila-Carbó (2009).

REFERENCES

Baccelli, F., Cohen, G., Olsder, G.J., and Quadrat, J.P. (1992). *Synchronization and Linearity: An Algebra for Discrete Event Systems.* Willey, New York.

Baccelli, F., Jean-marie, A., and Liu, Z. (1993). A survey on solution methods for task graph models. INRIA Report, Sophia Antipolis, France.

Guérin, R. and Orda, A. (2002). Computing shortest paths for any number of hops. *IEEE/ACM Trans. Netw.*, 10(5), 613–620.

Hernández-Orallo, E. and Vila-Carbó, J. (2009). Web server performance analysis using histogram workload models. *Computer Networks*, 53(15), 2727–2739.

Muller, A. and Stoyan, D. (2002). *Comparison Methods for Stochastic Models and Risks.* Wiley, New York, NY.

O'Connor, D. (2006). Exact and approximate distributions of stochastic PERT networks. Dublin University College.

Pekergin, N. and Vincent, J.M. (1991). Stochastic bounds on execution times of parallel programs. *IEEE Trans. Software Eng.*, 17(10), 1005–1012.

Tancrez, J.S., Semal, P., and Chevalier, P. (2009). Histogram based bounds and approximations for production lines. *European Journal of Operational Research*, 197(3), 1133–1141.