Deterministic estimation of the expected makespan of a POS under duration uncertainty

Michele Lombardi, Alessio Bonfietti, and Michela Milano

DISI, University of Bologna
{michele.lombardi2,alessio.bonfietti,michela.milano}@unibo.it

Abstract. This paper is about characterizing the expected makespan of a Partial Order Schedule (POS) under duration uncertainty. Our analysis is based on very general assumptions about the uncertainty: in particular, we assume that only the min, max, and average durations are know. This information is compatible with a whole range of values for the expected makespan. We prove that the largest of these values and the corresponding "worst-case" distribution can be obtained in polynomial time and we present an $O(n^3)$ computation algorithm. Then, using theoretical and empirical arguments, we show that such expected makespan is strongly correlated with certain global properties of the POS, and we exploit this correlation to obtain a linear-time estimator. The estimator provides accurate results under a very large variety of POS structures, scheduling problem types, and uncertainty models. The algorithm and the estimator may be used during search by an optimization approach, in particular one based on Constraint Programming: this allows to tackle a stochastic problem by solving a dramatically simpler (and yet accurate) deterministic approximation.

1 Introduction

A Partial Order Schedule (POS) is an acyclic graph $G = \langle A, E \rangle$, where A is a set of activities a_i, and E is a set of directed edges (a_i, a_j) representing end-to-start precedence relations. A POS is a flexible solution to a scheduling problem: some of the edges derive from the original Project Graph, while the remaining ones are added by an optimization approach so as to prevent potential resource conflicts.

A POS is very well suited as a solution format in the presence of duration uncertainty. Before the execution starts, each activity has a candidate start time. During execution, whenever a duration becomes known, the start times are updated (in $O(n^2)$ time) so that no precedence relation is violated: this guarantees that the schedule remains resource feasible.

In the presence of duration uncertainty, the quality of a POS should be evaluated via a stochastic metric, which in this paper is the expected value of the makespan: this is far from being a perfect choice, but still a fair one in many settings (in particular when the POS should be repeatedly executed, such as in stream processing applications).

A POS can be obtained essentially in two ways. First, in a constructive fashion, by adding edges to the graph until all potential resource conflicts are resolved: this is done by Precedence Constraint Posting (PCP) approaches [13, 7, 12, 10]. Alternatively, one can obtain a POS from a traditional schedule with fixed start times via a post-processing step [13, 12, 7, 8].

Despite the fact that a POS can adapt to uncertain durations, in practice the uncertainty is often largely disregarded when the graph is built. For example, all the post-processing methods operate on a schedule obtained for fixed durations. The PCP methods are capable of dealing with uncertainty, for example via scenario-based optimization or via the Sample Average Approximation [14, 5]: both approaches require to sample many duration assignments and to optimize a meta-model, resulting from the combination of a set of models (one per scenario) connected by chaining constraints. Despite this, most PCP approaches in the literature either assume to have fixed durations (e.g. [13, 7, 12]) or rely on minimal uncertainty information to perform robust (rather than stochastic) optimization (e.g. Simple Temporal Networks with Uncertainty [15, 11] or the method from [10]).

One of the main reasons for ignoring the uncertainty when searching for an optimal POS is the additional complexity. In fact, the complexity of a scenario-based approach depends on the number of samples required to have a satisfactory accuracy, which can be large for problems with many activities. The second main reason is the lack reliable information, either because data has not been collected or because the POS must be executed in a very dynamic environment.

In this paper, we tackle the problem of estimating the expected makespan of a POS under duration uncertainty. We use “deterministic” estimators rather than sampling-based statistics. The goal is providing an efficient alternative to scenario-based optimization in case the number of required samples is too high. Moreover, our estimators rely on very general assumptions on the uncertainty (in particular, we do not require independent durations) and are therefore well suited for cases in which extensive information is not available.

Technically, we focus on computing or approximating the largest value of the expected makespan that is compatible with the available uncertainty information. Our main contributions are: first, a proof that such a expected makespan value can be obtained in polynomial time and a $O(n^3)$ computation algorithm; second, a mixed theoretical/empirical analysis that highlights interesting properties of POSs and enables the definition of a linear-time approximate estimator. We have evaluated our estimator under an impressive variety of settings, obtaining reasonably accurate and robust results in most cases.

Our algorithm could be used within a heuristic scheduling method for estimating the value of the expected makespan. Alternatively, with further research the algorithm could be turned into a propagator for a global constraint, despite this is a non-trivial task. Our estimator consists instead of a non-linear formula that can be embedded in a CP approach using standard building blocks (sum and min operators): the resulting CP expression is propagated in $O(n)$. Indeed, this research line stems in part from the unexpectedly good performance of a

PCP approach based on Constraint Programming for robust scheduling, that we presented in [10]: our estimator in particular is designed to be used in a similar framework.

This paper is a follow-up of a previous work of ours [2], where we reached similar (although less mature) conclusions under more restrictive assumptions. The algorithm for computing the expected makespan is presented in Section 2, the analysis and the estimator are in Section 3, and Section 4 provides some concluding remarks.

2 Expected Makespan with Worst-Case Distribution

Our analysis is based on very general assumptions about the uncertainty. In particular, we assume that only the minimal, maximal, and average durations are known. Formally, the duration of each activity a_i can be modeled as a random variable D_i ranging over a known interval $[\underline{d}_i, \overline{d}_i]$ and having known expected value $\tilde{d}_i$. Note that the D_i variables are *not* assumed to be independent, and therefore they are best described in terms of their joint probability distribution.

Assuming that lower completion times are desirable, the makespan for a given instantiation of the D_i variables can be obtained by computing the longest path in G (often referred to as *critical path*). Formally, the makespan is a function $T(D)$ of the duration variables, where we refer as D to the vector of all D_i. The makespan is therefore itself stochastic, and $E[T(D)]$ denotes its expected value.

Because we rely on so little information about the uncertainty, the expected makespan is not unambiguously defined: the value of $E[T(D)]$ depends on the joint distribution of the D_i, and *there exists an infinite number of distributions that is compatible with known values of $\underline{d}$, $\overline{d}$, and $\tilde{d}$.*

The expected makespan cannot be higher than $T(\overline{d})$, i.e. than the makespan with maximal durations. Moreover, since the expected value is a linear operator, $E[T(D)]$ cannot be lower $T(\tilde{d})$, i.e. the makespan with expected durations.

Reaching stronger conclusions requires more powerful analysis tools, which will be presented in this paper. In particular, we are interested in the largest possible value of $E[T(D)]$ that is compatible with the known duration data: in absence of more information, this allows to obtain a safe estimate. The computation requires to identify a "worst-case" distribution, among the infinite set of distributions that are compatible with the given $\underline{d}$, $\overline{d}$, and $\tilde{d}$.

Properties of the Worst-case Distribution A generic (joint) probability distribution can be defined as a function:

$$P : \Omega \rightarrow [0, 1] \tag{1}$$

where Ω is a set of scenarios ω_k, each representing in our case an assignment for all D_i variables. The set Ω is called the *support* of the distribution and can have infinite size. The value $P(\omega_k)$ is the probability of scenario ω_k. The integral (or the sum, for discrete distributions) of P over Ω should be equal to 1. Our method for defining the worst-case distribution relies on a first, very important, result that is stated in the following theorem:

Theorem 1. *There exists a worst-case distribution such that, in each scenario with non-zero probability, every D_i takes either the value $\underline{d}_i$ or the value $\overline{d}_i$.*

The proof is in Appendix A. From Theorem 1 we deduce that it is always possible to define a worst-case distribution with a support Ω that is a subset of the Cartesian product $\prod_{a_i \in A} \left\{\underline{d}_i, \overline{d}_i\right\}$, and the therefore with size bounded by $2^{|A|}$.

Finding the worst-case distribution This makes it possible to model the construction of a worst-case distribution as an optimization problem. Formally, let us introduce for each scenario a decision variable $p_{\omega_k} \in [0,1]$ representing the value of $P(\omega_k)$. The assignment of the p_{ω_k} variables must be such that the known values of the expected durations are respected:

$$\overline{d}_i \sum_{\omega_k : D_i = \overline{d}_i} p_{\omega_k} + \underline{d}_i \left(1 - \sum_{\omega_k : D_i = \overline{d}_i} p_{\omega_k}\right) = \tilde{d}_i \text{ and hence: } \sum_{\omega_k : D_i = \overline{d}_i} p_{\omega_k} = \frac{\tilde{d}_i - \underline{d}_i}{\overline{d}_i - \underline{d}_i} \tag{2}$$

Then a worst-case distribution can be found by solving:

$$\mathbf{P0}: \quad \max z = \sum_{\omega_k \in \Omega} p_{\omega_k} T(D(\omega_k)) \tag{3}$$

$$\text{subject to: } \sum_{\omega_k : D_i = \overline{d}_i} p_{\omega_k} \leq \frac{\tilde{d}_i - \underline{d}_i}{\overline{d}_i - \underline{d}_i} \qquad \forall a_i \in A \tag{4}$$

$$\sum_{\omega_k \in \Omega} p_{\omega_k} \leq 1 \tag{5}$$

$$p_{\omega_k} \geq 0 \qquad \forall \omega_k \in \Omega \tag{6}$$

where $\Omega = \prod_{a_i \in A} \left\{\underline{d}_i, \overline{d}_i\right\}$. Equation (3) is the makespan definition ($D(\omega_k$ are the durations in ω_k), Equation (4) corresponds to Equation (2), and Equation (5) ensures that the total probability mass does not exceed one. It is safe to use a $\leq$ sign in Equation (4) and (5), because increasing a p_{ω_k} can only improve the problem objective and therefore all constraints are tight in any optimal solution. P0 is linear, but has unfortunately an exponential number of variables.

Tractability of P0: Luckily, P0 has a number of nice properties that make it much easier to solve. At a careful examination, one can see that P0 is the linear relaxation of a multi-knapsack problem. Results from LP duality theory [4] imply that the optimal solution cannot contain more than $\min(n, m)$ fractional variables, where n is the number of variables ($|\Omega|$ in our case) and m is the number of knapsack constraints (i.e. $|A| + 1$).

Due to Constraint (5), in an optimal solution of P0: either 1) all variables are integer and there is a single $p_{\omega_k} = 1$; or 2) all non-zero variables are fractional. Therefore, the number of non-zero variables in an optimal solution is bounded by $|A| + 1$. This means that a worst-case distribution with a support of size at most $|A| + 1$ is guaranteed to exist.

Algorithm 1 Compute $E_{wc}[T(D)]$

```
Require: A POS $G = \langle A, E \rangle$, plus $\underline{d}_i$, $\overline{d}_i$, and $\tilde{d}_i$ for each activity
 1: Let $T = 0$
 2: Let $s_{tot} = 1$ and let $s_i = (\tilde{d}_i - \underline{d}_i)/(\overline{D_i} - \underline{d}_i)$ for each $a_i \in A$
 3: while $s_{tot} > 0$ do
 4:     for all activities $a_i \in A$ do
 5:         Let $d_i = \overline{d}_i$ if $s_i > 0$, otherwise let $d_i = \underline{d}_i$
 6:     Find the critical path $\pi$ over $G$, using the $d_i$ values as durations.
 7:     Let $p_\pi = \min\left\{s_i : a_i \in \pi \text{ and } d_i = \overline{d}_i\right\}$
 8:     If $p_\pi = 0$, then $p_\pi = s_{tot}$
 9:     for all activities $a_i$ on the critical path such that $d_i = \overline{d}_i$ do
10:         Set $s_i = s_i - p_\pi$
11:     Set $s_{tot} = s_{tot} - p_\pi$
12:     Set $T = T + p_\pi \sum_{a_i \in \pi} d_i$
13: return $T$
```

Furthermore, all the variables of P0 appear with identical weights in all knapsack constraints (i.e. Equation (4) and (5)). Therefore, P0 can be solved to optimality by generalizing the solution approach for the LP relaxation of the classical knapsack problem: namely, one has to repeatedly: 1) pick the variable p_{ω_k} with the highest reward $T(D(\omega_k))$ in the objective; and 2) increase the value of p_{ω_k} until one of the knapsack constraints becomes tight.

A Polynomial Time Algorithm: We present (in Algorithm 1) a practical method for the computation of $E\left[T(D)\right]$ with worst-case distribution, i.e. $E_{wc}[T(D)]$. The algorithm improves the basic process that we have just described by relying on *partial scenarios*. A partial scenario represents a group of scenarios having the same makespan, and it is identified by specifying a duration (either $\underline{d}_i$ or $\overline{d}_i$) for a subset of the activities. Algorithm 1 keeps (line 2) a global probability budget s_{tot} and a vector probability budgets s_i to keep track of the slack in Constraint (5) and Constraints (4), respectively.

Then the algorithm repeatedly identifies a partial scenario that is compatible with the remaining slack and has maximal makespan. This is done by: 1) Assiging to each activity a duration d_i. This is equal to $\overline{d}_i$ is the corresponding slack variable s_i is non-zero, otherwise, the duration is $\underline{d}_i$ (lines 4-5); and 2) finding the critical path π on the POS, using the durations d_i (line 6).

At each iteration, the partial scenario is specified by: 1) the set of activities *on the critical path* that have maximal duration $\overline{d}_i$; and 2) the set of all activities for which $d_i = \underline{d}_i$. The idea is that if both sets of activities take their prescribed duration, then the critical path is π. This process always identifies the partial scenario that is compatible with the current slack and has the largest makespan.

The partial scenario is then inserted in the joint distribution with its largest possible probability: this is determined (with an exception discussed later) by smallest slack of the activities for which $d_i = \overline{d}_i$ in the partial scenario. The probability p_π of the partial scenario is used to update the slack variables at lines 9-11. Finally, the expected makespan variable T is incremented by the probability of the partial scenario, multiplied by its associated makespan (i.e. the sum of durations of the activities on the critical path).

The mentioned exception is that the probability p_π computed at line 7 may be 0 in case all actitivities on π have $d_i = \underline{d}_i$. If this happens, it means that the probability of the current partial scenario is not limited by Constraints (4), but only by Constraint (5). Hence, the scenario probability can be set to the value of the remaining global slack (line 8). When the global slack becomes zero, the algorithm returns the final value of $E_{wc}[T(D)]$.

By construction, at each iteration a slack variable s_i becomes zero. If all the s_i become zero, then the next partial scenario will have a critical path with only "short" activities and the probability update at line 8 will trigger. Overall, the algorithm can perform at most $|A| + 1$ iterations, in agreement with the theoretical prediction made for problem P0. The complexity of each iteration is dominated by the $O(|A|^2)$ longest path computation. The overall complexity is therefore $O(|A|^3)$, which is a remarkable result given that in principle the worst-case distribution had to be found among a set with infinite size.

3 Estimating $\mathbf{E_{wc}[T(D)]}$

Algorithm 1 allows an exact computation of the expected makespan with worst case distribution, but its scalability is limited by the $O(n^3)$ complexity. Moreover, the added difficulty of embedding a procedural component within an optimization approach may make the method less applicable in practice.

With the aim to address such limitations, we have devised a low-complexity, approximate, estimator based on global graph properties that are reasonably easy to measure. In detail, the estimator is given by the following formula:

$$\tau(\underline{T}, \overline{T}) = \underline{T} + \left(\overline{T} - \underline{T}\right) \frac{1}{|A|} \sum_{a_i \in A} \min\left(1, \frac{\tilde{d}_i - \underline{d}_i}{\overline{d}_i - \underline{d}_i} \frac{\sum_{a_i \in A} \overline{d}_i}{\overline{T}}\right) \tag{7}$$

Most of the terms in Equation (7) are constants that can be computed via a preprocessing step. The only input terms that change at search time are $\underline{T}$ and $\overline{T}$, which are simplified notations for $T(\underline{d}_i)$ and $T(\overline{d}_i)$: both values can be efficiently made available in a PCP approach via (e.g.) the methods from the Constraint Programming approach in [10]. The estimator formula is non-linear, but it is not difficult to embed in a CP model by using widely available building blocks: the resulting encoding is propagated in $O(|A|)$. The estimator is obtained via a non-trivial process that is discussed in detail in the remainder of this section. Its accuracy and robustness are discussed in Section 3.3.

3.1 A Simplified POS Model

The estimator from Equation (7) is obtained by reasoning on a simplified POS model. Specifically, such model is a layered graph with constant width, and identical minimal duration $\underline{d}_i = \underline{\delta}$ and maximal duration $\overline{d}_i = \overline{\delta}$ for each activity.

A layered graph is a directed acyclic graph whose activities are organized in layers: there are edges between each activity in k-th layer and each activity in the (k+1)-th layer, and no edge between activities in the same layer. In a layered graph with constant width, all layers contain the same number of activities.

The rationale behind our simplified model is that, when a POS is constructed, new edges are added to the original graph so as to prevent possible resource conflicts. This process leads to an "iterative flatting" of the POS (see [3]), making it closer in structure to a layered graph with constant width. The assumption on the identical minimal and durations is instead introduced only to increase the tractability.

Evaluating the Model: The effectiveness of using a layered graph as a model for more complex POSs is difficult to assess in a theoretical fashion, but can be checked empirically. Basically, if the model works, it should lead to reasonably accurate predictions of (e.g.) the value of $E[T(D)]$ with worst-case distribution.

This evaluation can be done by: 1) generating graphs with different number of activities, different width values, and different durations parameters; then 2) using Algorithm 1 to obtain the expected makespan with worst-case distribution. Then the results should compared with the results obtained for a set of POSs representing solutions to real scheduling problems: if the results are similar, it will be an indication of the model quality. In the remainder of this section, we will use this approach to investigate the impact of changing the width on the value of $E_{wc}[T(D)]$.

Measuring the Width: The width w of a layered graph is (by definition) the number of activities in each layer. The width of a real POS is instead a much fuzzier concept. In our evaluation, an approximate width value is computed as:

$$w = \frac{|A|}{n_L} \qquad \text{with: } n_L = \frac{\overline{T}}{avg(\overline{d}_i)} \qquad \text{and: } avg(\overline{d}_i) = \frac{1}{|A|}\sum_{a_i \in A} \overline{d}_i \tag{8}$$

Intuitively, we estimate the width as the ratio between the number of activities $|A|$ and the (estimated) number of layers n_L. The number of layers is obtained by dividing the makespan with maximal durations $\overline{T}$ by the average $\overline{d}_i$. By applying algebraic simplifications, we obtain the compact formula:

$$w = \frac{\sum_{a_i \in A} \overline{d}_i}{\overline{T}} \tag{9}$$

We use the same symbol (i.e. w) for both layered graphs and real POSs, because Equation (9) returns the "true" width when applied to our simplified model.

Evaluation Setup We generated layered graphs with different numbers of activities, and for each one we varied the width from $|A|$ (single layer) to 1. In order to obtain data points for every (integer) value of w, we allow for graphs with quasi-constant width: namely, if w is not a multiple of $|A|$, the right-most layers are permitted to contain $w - 1$ activities.

As a “real” counterpart for our empirical evaluation we employ a large benchmark of POSs representing solutions of Resource Constrained Project Scheduling Problems (RCPSP) and Job Shop Scheduling Problems (JSSP). The POSs for the RCPSP have been obtained by solving the j30, j60, and j90 instances from the PSPlib [6], respectively having $|A| = 30, 60, 90$. The POSs for the JSSP have been obtained by solving the instances in the Taillard benchmark, from size 15×15 to 30×20: in this case the number of activities ranges from 225 to 600.

Both the RCPSP and the JSSP instances have been solved with a classical CP approach [1], the SetTimes search strategy [9], and a makespan minimization objective. The POS have been obtained via the post-processing algorithm from [13] from all the schedules found during the optimization process, so that our collection contains 5,753 POSs coming from optimal, slightly suboptimal, and quite far from optimal schedules.

The instances in the PSPLIB and the Taillard benchmarks are originally deterministic, so we had to introduce some uncertainty artificially. In particular, the problem files specify a fixed duration for each activity, that we treat as the maximal duration $\overline{d}_i$. The values of $\underline{d}_i$ and $\tilde{d}_i$ are instead generated at random.

In this particular experimentation, the minimal duration of all activities was fixed to 0 for both the real POSs and the layered graphs. The maximum duration is fixed to 1 for the layered graphs (more on this later). All the average durations $\tilde{d}_i$ were randomly generated using a large variety of different schemes (see Section 3.3 for more details).

Makespan Normalization: Intuitively, the value of $E_{wc}[T(D)]$ should depend on the graph structure, on the characteristics of the uncertainty, and on the scale of the instance. Factoring out the dependence on the scale is important in order to understand the dependence on the structure and on the uncertainty. After some attempts, we have found that is possible to normalize the scale of the expected makespan without introducing distortions by using this simple formula:

$$\text{normalized}\left(E_{wc}[T(D)]\right) = \frac{E_{wc}[T(D)]}{\sum_{a_i \in A} \overline{d}_i} \tag{10}$$

i.e. by dividing the original value by the sum of the maximal durations. This normalization allows to compare on a uniform scale the expected makespan of graphs with widely different durations and number of activities. For our simplified model, the calculation makes the value of $\overline{d}$ completely irrelevant, which is the reason for fixing the maximal duration to 1 in the experimentation.

Some Results: We have obtained plots reporting the value of $E_{wc}[T(D)]$ over the graph width. Figure 1 shows the plots for the simplified model (with 64 activities) on the left, and for all the real POSs on the right. The values of $\tilde{d}_i$ in both cases have been generated uniformly at random between $\underline{d}_i$ (which is always 0) and 0.5 $\overline{d}_i$. The colors represent the data point density, which is benchmark dependent and unfortunately not very informative.

The results are very interesting. First, the general shape of the plots does not appear to depend on the number of activities: this can be inferred by the

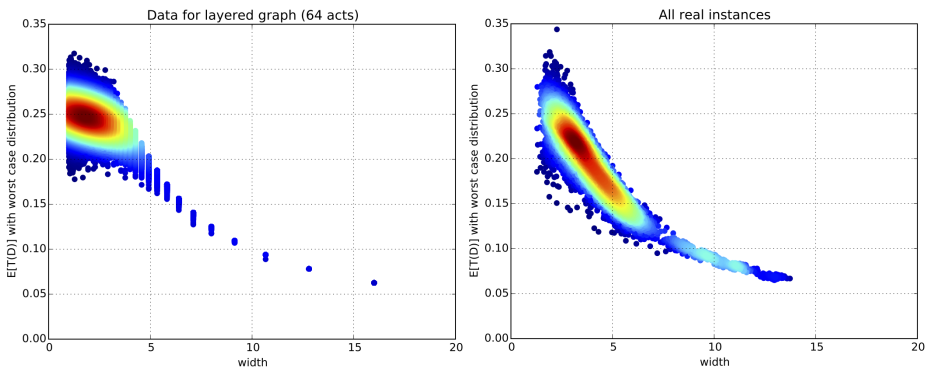


Fig. 1. Value of $E_{wc}[T(D)]$ over the graph width for the simplified model (left) and the real POSs (right)

plot for the real POSs, for which the number of activities ranges from 30 to 600. For the simplified model the plots with 16 and 32 activities (not reported) are almost indistinguishable from the plot with 64.

The most important result, however, is that the behavior of the simplified model and of the real graphs is remarkably similar, in terms of both shape and scale. This is particularly striking given the wide range of graph structures contained in our benchmark, and suggests that our simplified model is indeed well suited for studying the behavior of real POSs.

A necessary condition for the similarity to show up is that the values of the expected durations $\tilde{d}_i$ must be obtained using the same random generation scheme. This is not surprising, as it simply means that the characteristics of the uncertainty change the degree by which the value of w affects the expected makespan: this aspect will be tackled in the next section.

3.2 Deriving the Estimator

In this section, we use a mix of theoretical and empirical arguments applied to the simplified model from Section 3.1 to derive the estimator from Equation (7). In general terms, we are looking for some kind of formula that, based on global properties of the instance and the schedule can approximately predict the value of $E_{wc}[T(D)]$. Thanks to the simplicity of the layered graph model, this may be doable via Probability Theory results.

Random POS Construction: Let us assume to have a set of A of activities, with the corresponding minimal, maximal, and expected durations $\underline{d}_i$, $\overline{d}_i$, and $\tilde{d}_i$. In our simplified model all minimal durations are equal to $\underline{\delta}$ and all maximal durations are equal to $\overline{\delta}$: however, in this discussion we will use the $\underline{d}_i, \overline{d}_i$ notation whenever possible to maintain a clearer analogy with general POSs.

Since we are looking for a formula that depends on global properties, we can assume to have access general information (e.g. the width or the number of layers), but not to the location of each activity within the graph[1]. Therefore, we will assume that the layer where each activity appears is determined at random. This is a stochastic process and we are interested in the expected value of $E_{wc}[T(D)]$ w.r.t. the random mapping: i.e., we are dealing with a double expectation in the form $E_{\text{map}}[E_{wc}[T(D)]]$.

Single-layer $E_{wc}[T(D)]$: As a first step, we will focus on the inner expectation, i.e. on the computation of $E\left[T(D)\right]$ with worst case duration. At this stage, we can assume the activity positions to be known so that the expected makespan can be obtained by running Algorithm 1 (or solving P0). Since adjacent layers are fully connected, processing each layer separately is equivalent to processing the POS as whole. Now, consider the problem objective in P0:

$$\max z = \sum_{\omega_k \in \Omega} p_{\omega_k} T(\omega_k) \tag{11}$$

For a single layer of the simplified model it can be rewritten as:

$$\max z = \underline{\delta} + (\overline{\delta} - \underline{\delta}) \sum_{\omega_k : \exists D_i = \overline{\delta}} p_{\omega_k} \tag{12}$$

where we have exploited the fact that $\forall a_i \in A : \underline{d_i} = \underline{\delta}, \underline{d_i} = \overline{\delta}$, and therefore the single-layer makespan is equal to $\underline{\delta}$ unless at least one activity takes maximal duration. Therefore, the whole problem can be rewritten for a single-layer as:

$$\mathbf{P1}: \quad \max z = \underline{\delta} + (\overline{\delta} - \underline{\delta}) \sum_{\omega_k : \exists D_i = \overline{\delta}} p_{\omega_k} \tag{13}$$

$$\text{subject to: } \sum_{\omega_k : D_i = \overline{d_i}} p_{\omega_k} \leq \frac{\tilde{d_i} - \underline{d_i}}{\overline{D_i} - \underline{D_i}} \qquad \forall a_i \in A' \tag{14}$$

$$\sum_{\omega_k \in \Omega'} p_{\omega_k} \leq 1 \tag{15}$$

$$p_{\omega_k} \geq 0 \qquad \forall \omega_k \in \Omega' \tag{16}$$

where A' is the set of activities in the layer and $\Omega' = \prod_{a_i \in A'} \{\underline{d_i}, \overline{d_i}\}$. As already mentioned, we have not replaced $\underline{d_i}$, $\overline{d_i}$ with $\underline{\delta}$, $\overline{\delta}$ unless it was strictly necessary to obtain an important simplification.

Problem P1 is simple enough to admit a closed form solution. In particular, all scenarios where at least one activity takes the maximal duration are symmetrical. Moreover, it is always worthwhile to increase the probability of such scenarios as much as possible. Therefore the solution of P1 is always given by:

$$z^* = \underline{\delta} + (\overline{\delta} - \underline{\delta}) \min\left(1, \sum_{a_i \in A'} \frac{\tilde{d_i} - \underline{d_i}}{\overline{D_i} - \underline{D_i}}\right) \tag{17}$$

where $(\tilde{d_i} - \underline{d_i}) / \left(\overline{D_i} - \underline{D_i}\right)$ is the probability that a_i takes maximal duration.

[1] If we had access to this, we could use Algorithm 1

Expectation w.r.t. the Random Mapping In our stochastic POS construction process, the choice of the activities in each layer is random. Therefore the value of $(\tilde{d}_i - \underline{d}_i) / \left(\overline{D_i} - \underline{D_i}\right)$ can be seen as a random variable in Equation (17):

$$z^* = \underline{\delta} + (\overline{\delta} - \underline{\delta}) \min\left(1, \sum_{k=0}^{w-1} Q_k\right) \tag{18}$$

and selecting an activity for the current layer is equivalent to instantiating a Q_k.

For the first instantiation, the probability to pick a certain value of v_k for Q_k is given by the number of activities in A such that $(\tilde{d}_i - \underline{d}_i)/(\overline{D_i} - \underline{D_i}) = v_k$. For subsequent instantiations the probabilities will be different, because an activity cannot be inserted in two positions in the POS. In probabilistic terms, the Q_k variables are non-independent, which complicates enormously the task of defining a probabilistic model.

We address this issue by simply disregarding the constraint and allowing activity duplication to occur. We have empirically checked the accuracy of this approximation, which appear to be very good in terms of expected value. This last simplification is enough to ensure that: 1) the Q_k variables are independent; and 2) the Q_k variables have the same discrete distribution, which is defined by the frequency of occurrence of each $(\tilde{d}_i - \underline{d}_i)/(\overline{D_i} - \underline{D_i})$ among the $a_i \in A$.

Having independent and identically distributed Q_k implies that the value of Equation (18) will be identical for all the layers. Therefore a approximate formula for $E_{\text{map}}[E_{\text{wc}}[T(D)]]$ of the random POS construction process is:

$$n_L\, E\left[\underline{\delta} + (\overline{\delta} - \underline{\delta}) \min\left(1, \sum_{k=0}^{w-1} Q_k\right)\right] \tag{19}$$

where we recall that $n_L = |A|/w$ is the number of layers.

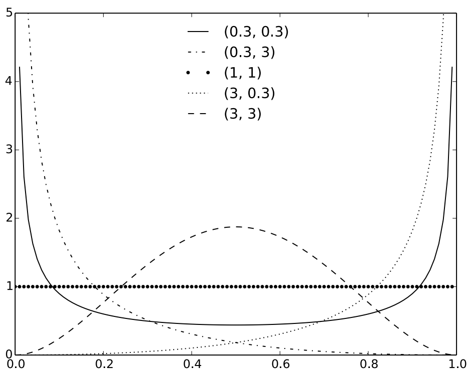
The Estimator Formula: Computing the expected value in Equation (19) is non-trivial, due to the presence of the "min" operator. It is however possible to address this issue by introducing some additional approximations. After exploring several alternatives, we have settled for replacing the sum of Q_k variables with a product. By exploiting the linearity of the expectation operator, we get:

$$n_L\, \underline{\delta} + n_L\, (\overline{\delta} - \underline{\delta}) E\left[\min\left(1, w\, Q_k\right)\right] \tag{20}$$

where it should be noted that $n_L\, \underline{\delta}$ and $n_L\, \overline{\delta}$ correspond for the simplified model to $\underline{T}$ and $\overline{T}$. Since Q_k has a discrete distribution, the expected value can be computed by: 1) multiplying by w every value in the distribution of Q_k; then 2) capping each result at 1; and finally 3) computing the average over all activities:

$$E\left[\min\left(1, wQ_k\right)\right] = \frac{1}{|A|} \sum_{a_i \in A} \min\left(1, w\, \frac{\tilde{d}_i - \underline{d}_i}{\overline{d}_i - \underline{d}_i}\right) \tag{21}$$

by combining Equation (20) and (21), replacing $n_L\, \underline{\delta}$ with $\underline{T}$, replacing $n_L\, \overline{\delta}$ with $\overline{T}$, and replacing w with the formula from Equation (9), we obtain the estimator original estimator formula.



It is useful to have a visual interpretation of the α and β parameters of the beta distribution. The plot on the left depicts five different shapes of the probability density function, which correspond to the configurations used in the experiments. The α parameter is intuitively related to the probability of the lower values, while β is related to the higher values. If $\alpha = \beta = 1$, all values are equally likely and we get a uniform distribution. If $\alpha < 1$ or $\beta < 1$ the probability mass tends to cluster respectively on the lowest and highest value. Conversely, if α or β are > 1, then the probability mass tends to move to the center, and the distribution assumes a bell-like shape.

Fig. 2. Visual interpretation of the α and β parameters in the beta distribution.

3.3 Empirical Evaluation of the Estimator

We have evaluated the estimator on the same benchmark used for assessing the effectiveness of the simplified model (see Section 3.1). We have compared the accuracy of our estimator $\tau(\underline{T}, \overline{T})$ with that of two baseline predictors: 1) the makespan with worst case durations $T(\overline{d})$, which is an upper bound for $E_{wc}[T(D)]$; and 2) and the makespan with expected durations $T(\tilde{d})$, which is a lower bound. In particular $T(\tilde{d})$ proved to be a surprisingly effective estimator for the expected makespan in a previous evaluation of ours [2].

We recall that the maximal durations for each POS are specified by the instance file. The minimal and the average durations are randomly generated, in this case using a beta distribution $Beta(\alpha, \beta)$. This was chosen because it is bounded and very flexible. The parameter configurations that we have employed are presented in Figure 3.3. In detail, the values of $\underline{d}_i$ and $\tilde{d}_i$ are obtained as:

$$\underline{d}_i = \frac{\overline{d}_i}{2} \, Beta(\alpha_{\underline{d}}, \beta_{\underline{d}}) \qquad\qquad \tilde{d}_i = \underline{d}_i + (\overline{d}_i - \underline{d}_i) \, Beta(\alpha_{\tilde{d}}, \beta_{\tilde{d}}) \tag{22}$$

where $\underline{d}_i$ ranges in $[0, \overline{d}_i/2]$, and $\tilde{d}_i$ ranges in $[\underline{d}_i, \overline{d}_i]$. We consider four parameter configurations for $\underline{d}_i$ and five configurations for $\tilde{d}_i$. Our goal was to generate a large variety of duration scenarios that are somehow representative of real world conditions. For instance the configuration $\alpha_{\tilde{d}} = 3, \beta_{\tilde{d}} = 3$ could model a situation where the activities are subject to many small disruptions. The configuration $\alpha_{\tilde{d}} = 0.3, \beta_{\tilde{d}} = 3$ leads to average durations very close to the minimal ones, which may be a model for unlikely, but very disruptive, machine faults.

Table 1 shows the average and standard deviation of the relative prediction error for the compared estimators. We use percentage errors computed as:

$$Err(x) = 100 \, \frac{E_{wc}[T(D)] - x}{E_{wc}[T(D)]} \tag{23}$$

where x is the prediction given by an estimator. Negative errors represent over-estimations, while positive errors are under-estimations. As expected, the $T(\overline{d})$ predictor always over-estimates, while $T(\tilde{d})$ under-estimates.

As the number of activities grows, $E_{wc}[T(D)]$ gets closer to $T(\overline{d})$: intuitively, since we are reasoning with a worst-case distribution, more activities mean more things that can go wrong. This trend reflects on the performance of $T(\overline{d})$ (which gets better), and appears to have a adverse effect on the accuracy of τ and $T(\tilde{d})$.

When the minimal durations tend to be higher (i.e. when $\alpha_{\underline{d}} \geq 1$ and $\beta_{\underline{d}} < 0$), the range of the random durations is reduced and all predictions are more precise. The $T(\tilde{d})$ estimator benefits the most from this situation.

In general, the τ estimator tends to outperform the other estimators: the accuracy is much better than $T(\overline{d})$, and significantly better than $T(\tilde{d})$ in most cases. It must be said that using $T(\tilde{d})$ as an estimator for $E_{wc}[T(D)]$ lead to surprisingly good results, especially in terms of standard deviation: this is consistent with our findings form [2], and deserves further investigation. The τ estimator really shines when the $\tilde{d}_i$ values are small, i.e. for $\alpha_{\tilde{d}} < 0$: this is the situation that models the presence of machine faults, which is also the best suited for reasoning in terms of worst-case distribution (since a fault tends to affect all the activities in the system).

4 Concluding Remarks

We have tackled the problem of estimating the expected makespan of a POS under duration uncertainty, assuming access to very limited information about the probability distribution. We have provided an $O(n^3)$ algorithm for computing the largest expected makespan that is compatible with the available information, and we have proposed a linear-time estimator to approximately predict the same quantity. Both the algorithm and the estimator could be employed within an optimization approach to obtain an estimate of the expected makespan, in cases where scenario-based optimization has prohibitive complexity, or extensive duration data is not available.

Our results are very encouraging, and we believe that further improvements are possible. First, it should be possible to exploit better the striking similarity of behavior between real POSs and layered graphs, which we observed in [2] and confirmed in this paper. Second, it may be possible to obtain a more accurate and still efficient approach by exploiting additional information (e.g. a covariance matrix). Third, we have largely disregarded the strong correlation between $T(\tilde{d})$ and the expected makespan, which we had observed in [2] and was confirmed in our experimental evaluation: it should be possible to exploit this correlation in order to improve the accuracy of the estimate.

Additional possible directions for future research involve the design of an estimator for the makespan variance, or for specific quantiles in its distribution. Such an estimator would allow to state “approximate chance constraints”, and allow an efficient solution of many more practical problems. Finally, it may be possible to replace our scalar estimate with ranges, similarly to what is done in statistics with confidence intervals. A the present time, we have not started to investigate any of the last two mentioned research directions.

5 Appendix A

This appendix contains the proof to Theorem 1.

Theorem 1. *There exists a worst-case distribution such that, in each scenario with non-zero probability, every D_i takes either the value $\underline{d}_i$ or the value $\overline{d}_i$.*

Proof (By contradiction and induction). Let us assume that the worst case distribution contains a scenario ω_k such that one D_i is equal to some value $v_k \in (\underline{d}_i, \overline{d}_i)$ and $P(\omega_k) > 0$. Let $\underline{\omega}_k$ and $\overline{\omega}_k$ be two scenarios that are identical to ω_k in all assignments, except that D_i takes respectively its minimal and maximal value. Let us consider shifting the probability $P(\omega_k)$ from ω_k to $\underline{\omega}_k$ and $\overline{\omega}_k$, and let $\Delta P(\underline{\omega}_k) \geq 0$ and $\Delta P(\overline{\omega}_k) \geq 0$ be the two probability variations. Then it must hold:

$$\Delta P(\overline{\omega_k}) + \Delta P(\underline{\omega_k}) = P(\omega_k) \tag{24}$$

in order to preserve the total probability mass, and:

$$\overline{d}_i \Delta P(\overline{\omega_k}) + \underline{d}_i \Delta P(\underline{\omega_k}) = v_k P(\omega_k) \tag{25}$$

to keep the expected value $\tilde{d}_i$ unchanged. We recall that in each scenario the makespan is given by the length of the critical path, and that the expected makespan is given by the integral (or sum) of $T(D(\omega_k))P(\omega_k)$. Now, let us distinguish some cases:

1) $\mathbf{a_i}$ is on the critical path in ω_k. This implies that a_i is on the critical path also in $\overline{\omega}_k$, and therefore shifting the probability causes the expected makespan to increase by $(\overline{d}_i - v_k)\Delta P(\overline{\omega}_k)$ units. Then:
- If a_i is on the critical path in $\underline{\omega}_k$, then the expected makespan is also decreased by $(v_k - \underline{d}_i)\Delta P(\underline{\omega}_k)$ units. However, by combining Equations (24) and (25) we obtain that:

$$(\overline{d}_i - v_k)\Delta P(\overline{\omega}_k) - (v_k - \underline{d}_i)\Delta P(\underline{\omega}_k) = 0 \tag{26}$$

  i.e. the net change of the expected makespan is zero.
- If a_i is *not* on the critical path in $\underline{\omega}_k$, then the expected makespan is decreased by less than $(v_k - \underline{d}_i)\Delta P(\underline{\omega}_k)$ units, and the net change is positive (i.e. the expected makespan is increased).

2) $\mathbf{a_i}$ is *not* on the critical path in ω_k. This implies that a_i is not on the critical path also in $\underline{\omega}_k$, and therefore that shifting probability to $\underline{\omega}_k$ leaves the expected makespan unchanged. Then:
- If a_i is on the critical path in $\overline{\omega}_k$, the expected makespan is increased by a quantity in the range $[0, (\overline{d}_i - v_k)\Delta P(\overline{\omega}_k))$.
- If a_i is *not* on the critical path in $\overline{\omega}_k$, the expected makespan is again unchanged.

Therefore, by reducing to zero the probability $P(\omega_k)$, the expected makespan either is increased or stays unchanged. This procedure can be repeated until there is no scenario with non-zero probability where D_i takes a value different from $\underline{d}_i$ or $\overline{d}_i$. □

Table 1. Partial horizontal line

J30

$(\mathbf{a}_{\tilde{\mathbf{d}}_\mathbf{i}}, \mathbf{b}_{\tilde{\mathbf{d}}_\mathbf{i}}) \longrightarrow$		**(0.3,0.3)**			**(0.3,3)**			**(1,1)**			**(3,3)**		
$(\mathbf{a}_{\underline{\mathbf{d}}_\mathbf{i}}, \mathbf{b}_{\underline{\mathbf{d}}_\mathbf{i}})$		τ	$\mathbf{T}(\tilde{\mathbf{d}})$	$\mathbf{T}(\bar{\mathbf{d}})$	τ	$\mathbf{T}(\tilde{\mathbf{d}})$	$\mathbf{T}(\bar{\mathbf{d}})$	τ	$\mathbf{T}(\tilde{\mathbf{d}})$	$\mathbf{T}(\bar{\mathbf{d}})$	τ	$\mathbf{T}(\tilde{\mathbf{d}})$	$\mathbf{T}(\bar{\mathbf{d}})$
(0.3,0.3)	**Err**	-1.08	5.06	-37.19	-3.91	9.37	-148.18	-7.75	11.32	-31.82	-12.23	15.87	-27.92
	Std	7.77	2.72	13.38	5.36	4.57	32.46	6.94	3.79	10.29	6.51	3.99	8.27
(0.3,3.0)	**Err**	1.55	7.54	-52.16	2.44	24.83	-352.00	-6.83	17.14	-41.85	-13.18	24.20	-35.18
	Std	10.07	3.87	19.37	6.77	8.80	113.50	8.32	4.87	13.75	8.06	4.61	10.57
(3.0,0.3)	**Err**	-1.62	2.66	-26.68	-3.22	3.54	-85.42	-6.38	6.22	-23.62	-9.90	9.33	-21.15
	Std	5.55	1.67	8.54	3.38	2.19	10.22	4.85	2.59	7.03	4.82	2.97	6.01
(1.0,1.0)	**Err**	-0.70	5.14	-37.30	-3.02	10.13	-153.98	-7.04	11.33	-32.04	-11.91	16.20	-27.98
	Std	7.71	2.66	13.60	4.79	4.52	30.28	6.42	3.64	10.17	6.28	3.88	8.10
(3.0,3.0)	**Err**	-0.03	5.29	-38.33	-2.17	10.25	-160.43	-6.95	11.54	-32.73	-11.72	16.46	-28.09
	Std	7.42	2.75	13.65	4.53	4.39	29.59	6.59	3.62	10.15	6.44	3.75	8.09
J60													
(0.3,0.3)	**Err**	-2.92	6.16	-33.85	-7.35	12.62	-133.52	-11.11	13.10	-27.88	-16.20	18.25	-23.59
	Std	6.66	2.45	10.35	5.12	4.32	23.12	5.81	3.25	7.72	5.33	3.38	6.15
(0.3,3.0)	**Err**	0.07	8.82	-44.82	-0.34	33.56	-268.74	-11.47	19.79	-35.42	-18.73	27.21	-29.12
	Std	8.10	3.39	14.08	6.30	7.83	66.40	7.54	4.06	9.63	6.62	3.68	7.42
(3.0,0.3)	**Err**	-3.19	3.32	-25.27	-6.40	4.93	-81.94	-9.31	7.54	-21.56	-13.37	10.95	-18.81
	Std	4.90	1.61	7.01	3.52	2.32	8.50	4.46	2.32	5.54	4.27	2.57	4.81
(1.0,1.0)	**Err**	-2.34	6.10	-33.65	-6.61	13.79	-139.56	-10.83	13.31	-28.08	-16.30	18.50	-23.87
	Std	6.57	2.47	10.04	4.82	4.53	22.94	5.77	3.21	7.57	5.50	3.32	6.21
(3.0,3.0)	**Err**	-1.84	6.22	-34.20	-5.66	14.53	-145.00	-10.48	13.52	-28.10	-16.10	18.69	-23.90
	Std	6.63	2.46	10.24	4.85	4.61	22.46	5.72	3.16	7.41	5.37	3.17	6.14
J90													
(0.3,0.3)	**Err**	-3.94	6.59	-32.30	-9.82	14.67	-125.92	-12.44	13.87	-26.23	-17.41	19.20	-22.03
	Std	6.28	2.35	9.03	5.20	4.16	19.42	5.43	2.92	6.54	4.93	3.17	5.54
(0.3,3.0)	**Err**	-1.16	9.62	-42.30	-1.25	38.04	-233.89	-13.54	20.81	-33.02	-20.17	28.48	-26.67
	Std	7.39	3.19	11.32	6.35	6.83	52.10	6.77	3.74	8.34	6.04	3.35	6.54
(3.0,0.3)	**Err**	-3.71	3.62	-24.17	-8.44	5.78	-79.91	-10.55	8.16	-20.61	-14.43	11.71	-17.88
	Std	4.59	1.55	6.08	3.70	2.42	8.03	4.26	2.25	5.03	3.88	2.51	4.24
(1.0,1.0)	**Err**	-3.46	6.67	-32.51	-8.80	15.97	-131.57	-12.31	14.23	-26.29	-17.24	19.42	-21.97
	Std	5.85	2.36	8.44	5.08	4.33	19.47	5.53	2.95	6.59	4.86	3.02	5.40
(3.0,3.0)	**Err**	-2.73	6.83	-32.88	-7.77	17.12	-136.15	-12.12	14.35	-26.59	-17.36	19.80	-22.28
	Std	5.84	2.36	8.55	5.07	4.54	19.52	5.47	2.91	6.63	5.00	2.92	5.47
Taillard													
(0.3,0.3)	**Err**	-7.34	11.16	-27.74	-14.69	27.53	-94.63	-12.85	20.99	-17.84	-12.29	26.59	-12.63
	Std	4.16	2.36	5.69	3.69	3.77	10.76	3.51	2.64	4.09	3.42	2.26	3.50
(0.3,3.0)	**Err**	-5.74	15.93	-34.45	-6.04	55.75	-134.53	-14.01	29.17	-20.96	-13.88	36.39	-14.33
	Std	4.66	3.04	6.73	4.14	3.54	17.82	4.07	2.97	4.90	3.69	2.43	3.79
(3.0,0.3)	**Err**	-6.97	6.92	-22.08	-13.61	13.95	-66.55	-11.20	13.85	-14.98	-10.62	17.87	-10.87
	Std	3.35	1.76	4.15	3.18	2.68	6.65	2.94	2.14	3.29	2.92	2.10	2.98
(1.0,1.0)	**Err**	-7.17	11.29	-28.19	-13.26	29.75	-96.61	-12.84	21.31	-17.97	-12.31	26.88	-12.65
	Std	3.95	2.44	5.27	3.73	3.46	11.00	3.44	2.54	4.00	3.36	2.34	3.46
(3.0,3.0)	**Err**	-6.49	11.45	-28.09	-11.98	31.37	-98.08	-12.72	21.43	-17.98	-12.28	27.13	-12.62
	Std	3.92	2.45	5.34	3.81	3.39	11.24	3.51	2.56	4.05	3.35	2.24	3.44

References

1. P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-based scheduling.* Kluwer Academic Publishers, 2001.
2. A. Bonfietti, M. Lombardi, and M. Milano. Disregarding duration uncertainty in partial order schedules? yes, we can! In *Proc. of CPAIOR*, pages 210–225, 2014.
3. A. Cesta, A. Oddi, and S.F. Smith. Iterative flattening: A scalable method for solving multi-capacity scheduling problems. In *AAAI/IAAI*, pages 742–747, 2000.
4. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems.* Springer Science & Business Media, 2004.
5. A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. The sample average approximation method for stochastic discrete optimization. *SIAM Journal on Optimization*, 12(2):479–502, 2002.
6. R. Kolisch and A. Sprecher. Psplib-a project scheduling problem library: Or software-orsep operations research software exchange program. *European journal of operational research*, 96(1):205–216, 1997.
7. P. Laborie. Complete MCS-Based Search: Application to Resource Constrained Project Scheduling. In *Proc. of IJCAI*, pages 181–186. Professional Book Center, 2005.
8. P. Laborie and D. Godard. Self-adapting large neighborhood search: Application to single-mode scheduling problems. In *Proc. of MISTA*, 2007.
9. C. Le Pape, P. Couronné, D. Vergamini, and V. Gosselin. Time-versus-capacity compromises in project scheduling. *AISB QUARTERLY*, page 19, 1995.
10. M. Lombardi, M. Milano, and L. Benini. Robust scheduling of task graphs under execution time uncertainty. *IEEE Trans. Computers*, 62(1):98–111, 2013.
11. P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *Proc. of IJCAI*, pages 494–499. Morgan Kaufmann Publishers Inc., 2001.
12. N. Policella, A. Cesta, A. Oddi, and S. F. Smith. From precedence constraint posting to partial order schedules: A CSP approach to Robust Scheduling. *AI Communications*, 20(3):163–180, 2007.
13. N. Policella, S. F. Smith, A. Cesta, and A. Oddi. Generating Robust Schedules through Temporal Flexibility. In *Proc. of ICAPS*, pages 209–218, 2004.
14. S. A. Tarim, S. Manandhar, and T. Walsh. Stochastic constraint programming: A scenario-based approach. *Constraints*, 11(1):53–80, 2006.
15. T. Vidal. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999.