Evaluation of the Quantiles and Superquantiles of the Makespan in Interval Valued Activity Networks

Carlo Meloni[1]

Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti", Sapienza Università di Roma, Roma (I)

email: `carlo.meloni@uniroma1.it`

Marco Pranzo

Dipartimento di Ingegneria dell'Informazione e Scienze Matematiche, Università di Siena, Siena (I)

email: `marco.pranzo@unisi.it`

Abstract

This paper deals with the evaluation of quantile-based risk measures for the makespan in scheduling problems represented as temporal networks with uncertainties on the activity durations. More specifically, for each activity only the interval for its possible duration values is known in advance to both the scheduler and the risk analyst. Given a feasible schedule, we calculate the quantiles and the superquantiles of the makespan which are of interest as risk indicators in various applications.

To this aim we propose and test a set of novel algorithms to determine rapid and accurate numerical estimations based on the calculation of theoretically proven lower and upper bounds. An extensive experimental campaign computationally shows the validity of the proposed methods, and allows to highlight their performances through the comparison with respect to the state-of-the-art algorithms.

Keywords: Quantiles, Superquantiles, VaR, Expected Shortfall, CVaR, Scheduling, Makespan, Uncertainty, Activity Networks

[1]Corresponding author.

1. Introduction

The reliability of a scheduling solution is important for practitioners to take informed decisions, and a tool able to quantify the risk associated to a schedule offers a practical support in the decision process. Therefore, the adoption of a suitable risk measure is an issue in the modeling scheduling problems arising in different applications including: task or project bidding [25], risk evaluation and mitigation [67, 68], due date setting or acceptance [5, 64], orders proposal and acceptance [34], robust planning or scheduling [2, 13, 19, 44], and solution ranking and selection [8].

Temporal activity networks are extensively adopted in various domains to model planning and scheduling problems with analysis and optimization purposes. In these contexts, considering the evaluation of the quantile and the superquantile of a performance index appears to be of interest to control both its expected value and variability [8, 10] while keeping into account the decision maker risk attitude. These risk indicators are called Value-at-Risk and Conditional Value-at-Risk or Expected Shortfall in different domains, but following [8, 59] we adopt an application-independent terminology.

In this paper we consider the makespan as the relevant performance index associated to the risk. Therefore we focus on the computational evaluation of the quantile (indicated as q) and the superquantile (denoted as Q) of the makespan. They are two popular risk indices which refer to a probability level α (e.g., see [8, 10, 59, 62]), hereinafter indicate as q_α and Q_α, respectively. The quantile q_α can be defined informally as the maximum possible loss (e.g., representing a delay in manufacturing, transportation or supply services) for all $\alpha 100\%$ best outcomes, i.e., it does not assess the magnitude of possible losses. Differently, Q_α measures the average loss in the worst $(1-\alpha)100\%$ of outcomes and emphasizes the contribution of the worst-case outcomes under the index performance of interest [8, 10, 59].

This research study proposes a set of new algorithms to evaluate both q_α and Q_α of the makespan of a given feasible solution for a scheduling problem represented by a temporal activity network [24, 61, 69] and given a probability level α. In this way the assessment of the risk for a given solution (in whatever way it is obtained) is performed. More specifically, according to [27, 49, 70, 71], we consider the class of scheduling models with a fixed and given set of precedence relations, but (independent) uncertain integer processing times belonging to known intervals of equally possible values.

The computation of both q_α and Q_α when the performance index is the makespan for a given activity network representing a schedule in general is known to be NP-complete [31]. To obtain an accurate and rapid evaluation, extending the approach recently proposed by [49], we develop a set of heuristic computational methods which are pseudo-polynomial in the general case. They are based on the calculation of two tight theoretical bounds for the considered risk indicators.

The main contributions of this paper can be schematically summarized in the following: *i*) it introduces a set of graph-based transformations and approximations as main components of the proposed computational methodology; *ii*) it proposes new *accurate* and *rapid* methods to calculate q_α and Q_α of the makespan (on the basis of theoretically proven lower and upper bounds) of a temporal network when the activity durations are represented by integer intervals of possible values; *iii*) it reports on a campaign of computational experiments conducted on an extensive set of benchmark instances and a comparison to state-of-the art algorithm known in the literature showing the effectiveness of the proposed algorithms; *iv*) it offers an experimental analysis to address also the algorithmic flexibility issues, and to investigate challenging instances.

The rest of the paper is structured as follows. Section 2 offers a literature review, while Section 3 introduces the considered temporal networks models for scheduling problems under uncertainty. In Section 4 the proposed algorithms for q_α and Q_α of the makespan are described also using an illustrative example. The extensive computational experiments are introduced and discussed in Section 5. Finally, conclusions and future research are drawn in Section 6.

2. Literature review

Scheduling problems involve the execution of a given set of activities on a finite number of resources with the aim to optimize given performance measures [11]. The latter are usually in the form of a scalar function of activities completion times, and the makespan is the most popular of these measures [11, 63].

Typically, scheduling issues are modelled as deterministic problems and therefore the resulting mathematical model may not be an adequate representation of an uncertain reality [6]. Even if this assumption may be valid when the effects of uncertainties are limited [40], it is not valid or acceptable for all real contexts. A typical consequence of disregarding activity duration uncertainties is the underestimation of the expected makespan frequently observed in practice. This phenomenon has been the subject of extensive research in services and manufacturing under both the operations research (e.g., see [33, 39, 53]) and the industrial management (e.g., see [41, 42, 46]) perspectives.

In the literature there is no universally accepted single risk measure considering that each of them has its own pros and cons [8, 10, 59].

Indeed, operation times may vary due to a change in a dynamic environment or to an information update, and one way to address this concern is to use stochastic models [6]. They are usually introduced for uncertain scheduling problems, where model elements such as specific operation times are assumed to be random variables with known probability distribution functions. However, these models typically require higher computational times that are not always compatible with stringent operational or real-time requirements [41, 61].

In these contexts, the evaluation of the risk associated to a performance index for a given schedule can be obtained taking advantage of a real-time (or at least updated) forecasting of duration times, assuming that they are represented by prediction intervals [47, 49]. In other real situations time intervals may represent a partial knowledge of the duration of the operations and a distributionally robust analysis is conducted [1, 4], or they represent an imprecise indication of their equally possible values [27, 52, 70, 71]. This makes the risk assessment particularly useful as in many scheduling applications the decision makers may be risk-averse and may prefer solutions that do not just perform well “on average”, but that also perform satisfactorily “in most cases” [19, 25, 26].

The variance of the makespan has been widely used in the literature as a risk measure [20, 61, 69]. However, this may not be always appropriate for optimization problems in temporal networks. Since when the goal is to minimize the makespan downward deviations are indeed desirable [10, 26], and may therefore prefer to consider one-sided quantile-based measures of the objective for stochastic optimization (e.g., see [10, 57], and [3, 62] for scheduling problems) as for instance q_α and Q_α. Usually, some scenario-based (or sample-based) method is adopted to estimate these risk measures for scheduling problems, and this approach can require a relevant computational effort and expose to estimation accuracy issues [3, 62, 65].

Assuming the index $\mathbf{P}$ (uncertain quantities are hereinafter represented in bold) is to be minimized, they can be more formally described:

- Quantile of $\mathbf{P}$ at probability level α ($q_\alpha(\mathbf{P})$):

$$q_\alpha(\mathbf{P}) = inf\{t : prob(\mathbf{P} \leq t) \geq \alpha\} \tag{1}$$

  It is a threshold that is exceeded in $(1-\alpha)100\%$ of all cases. It represents the so called Value-at-Risk of $\mathbf{P}$ at probability level α.

- Superquantile of $\mathbf{P}$ at probability level α ($Q_\alpha(\mathbf{P})$):

$$Q_\alpha(\mathbf{P}) = E(\mathbf{P}|\mathbf{P} \geq q_\alpha(\mathbf{P})) \tag{2}$$

  $\mathrm{Q}_\alpha(\mathbf{P})$ is the expected value of all cases exceeding the threshold $\mathrm{q}_\alpha(\mathbf{P})$, and corresponds to the Expected Shortfall or the Conditional-Value-at-Risk at level α of $\mathbf{P}$ [10, 58]:

$$\mathrm{Q}_\alpha(\mathbf{P}) = \frac{1}{1-\alpha} \int_\alpha^1 \mathrm{q}_\beta(\mathbf{P})\, d\beta \tag{3}$$

Considering these assumptions and definitions, $\mathrm{q}_\alpha(\mathbf{P}) \leq \mathrm{Q}_\alpha(\mathbf{P})$ holds for all $\alpha \in [0, 1]$.

Our analysis focuses on both these quantile-based risk measures which provide in general enough flexibility to design decision methods that are more discerning than procedures that use the expected values [8]. Nevertheless, when a decision maker is not only concerned with the frequency of undesirable outcomes, but also with their severity, Q_α is recommended instead of q_α [10, 62]. In this case higher values of α are chosen by decision makers who are more risk-averse, and $\alpha = 0$ represents the risk-neutral choice. In fact, as α tends to 1 (i.e., its upper extremum), both $\mathrm{q}_\alpha(\mathbf{P})$ and $\mathrm{Q}_\alpha(\mathbf{P})$ tend to the worst or pessimistic case value $\overline{P}$; while when α tends to 0, $\mathrm{q}_\alpha(\mathbf{P})$ tends to the best or optimistic value $\underline{P}$, whereas $\mathrm{Q}_\alpha(\mathbf{P})$ tends to the expected value of the performance index $E(\mathbf{P})$.

Although the use of quantile-based measures to take into account uncertainty in scheduling problems seems a promising topic, only few relevant papers appeared in the literature. They are mainly related to some specific application [14, 30, 35, 60], or limited to some specific scheduling environment [3, 17, 62]. Compared to other sectors, where both quantiles and superquantiles are more used, their restricted diffusion in scheduling is perhaps due to the computational difficulties connected to the adopted approaches that are often based on scenarios and sampling methods [8, 40, 61, 62], which lead to limiting the analysis to scheduling cases with simple layout configurations, offering as output statistical estimates of the indicators of interest [12, 43, 62].

The computation of the expected shortfall of the makespan in scheduling problems represented by activity networks where a reliable prior knowledge of random processing times is not available has been recently addressed in [49] considering cases in which the processing times are integers and their equally possible values belong to known intervals (i.e., only a given number of integer values are possible for each duration time). These cases can be considered as scheduling problems with incomplete or imperfect information [15, 16, 27, 39, 70, 71], assuming that the realized duration of an activity is only known when the execution of the activity is completed, and the uncertainty on these parameters implies that the objective value is also uncertain. These quantile-based risk assessment issues can arise in different real contexts including project management [49], industrial scheduling [51] and transportation [50] requiring fast and accurate computation methods. On the basis of the availability of these methods, different application schemes can be considered enabling further studies and researches, the main being:

- *Solution risk assessment.* In this case the considered methods are used to evaluate the risk associated with a known schedule represented by a specific activity network. This schedule represents the result of a previous selection or optimization process and the risk assessment could be part of a validation or pre-implementation phase, or even of a final (possible multi-criteria) selection among several candidate schedules [25]. In this case there are, in general, no severe computation time restrictions and the methods can pursue exact or very accurate calculations.

- *Constructive scheduling algorithms.* In this case the methods can be used to implement a score function to guide either heuristic or exact constructive scheduling algorithms [54, 55] which can also be structured in a metaheuristic scheme [48, 56]. These optimization approaches require, in general, fast scoring functions.

- *Iterative or improving scheduling algorithms.* In exact and heuristic (or metaheuristic) iterative algorithms [28, 54] the methods can be used as main (or secondary) score functions or as constraints to take into account the decision maker's preferences. These schemes are usually (yet not always) based on the local search paradigm, and also in this case fast scoring functions are in general preferred.

- *Simulation-optimization scheduling algorithms.* Approaches that offer accurate estimates of quantile-based measures of the makespan and that are faster than simulation or scenario-sampling methods ([61, 62]) can be advantageously employed in simulation-optimization scheduling schemes [45].

- *Risk monitoring and re-scheduling.* The methods can be also used to monitor the risk during the execution of the activities. In fact, the actual value of the duration of an activity is known only at the end of it. Thus, the risk assessment for a schedule can be updated at the end of an activity or at a significant event (i.e., considered as a milestone). This updating activity can be limited to a monitoring service or more usefully can trigger a re-scheduling [40, 66] of the activities not yet executed. Though computation time is not always a hard constraint for monitoring, however, generally, the faster the better.

3. Integer Interval-Valued Activity Networks

Temporal activity networks (TANs) are a widely used modeling tool for time management of complex systems or projects [24, 69]. In its basic form, a TAN is assumed to have deterministic activity durations (or times), and can be referred to as a deterministic temporal activity network (DTAN). These networks are in general relatively easy to analyze with respect to the makespan often involving some *longest path problem* [24, 36]. However, for real problems, activity durations are unlikely to be deterministic and their values can be affected by several factors. Stochastic temporal activity networks (STANs) extend these models in presence of uncertainties, and are a suitable modelling tool for problems that could not be adequately represented by deterministic networks.

According to [24, 69], we represent the precedence constraints by a directed acyclic graph (DAG) $G = (N, A)$, called the precedence graph. More specifically, we adopt the so called *Activity-on-Arcs* (AoA)[22] representation having nodes $N = \{1, \dots, n\}$ representing events, and arcs $A \subseteq \{(i, j) : i, j = 1, \dots, n\}$ representing both activities and precedence constraints. An arc $(i, j) \in A$ means that the event associated to node j can occur only after event i. Without loss of generality, we consider networks having one initial node (source) and one ending node (sink). The nodes in N are topologically labeled in such a way that $i < j$ for each arc $(i, j) \in A$. Thus, initial and end nodes have labels 1 and $n = |N|$, respectively.

A DTAN is a DAG along with a duration for each arc, i.e., a pair (G, d), where d contains all the deterministic values $d(i, j) \geq 0$, where an arc representing a generalized logic precedence could have associated a null duration [23]. We assume, as usual in scheduling problems [6, 11], that the durations have integer values, i.e., they represent an integer number of a suitable elementary time unit.

A STAN is an activity network with random activity durations, and can be described by the pair $(G, \mathbf{D})$, where $G = (N, A)$ is the precedence DAG, and $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$ is the vector of $m = |A|$ random durations associated to the arcs. In a STAN, all quantities that depend on the activity durations are random variables. Therefore, the starting and completion time of any activity, and the makespan are all random variables. We denote the makespan (random variable) as $\mathbf{C}_{max}$.

Following [49] we consider particular STANs referred to as *Integer Interval-Valued Activity Networks* (IIN) in which interval activity integers times $T_i = [\underline{t_i}, \overline{t_i}]$ for $i = 1, \ldots, m$ are assigned to the activity durations $\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$ in the network and represent the equally possible durations expressed in terms of the number of a specific time units. The values $\underline{t_i}$ and $\overline{t_i}$ are integers with $\underline{t_i} \geq 0$, and $\underline{t_i} \leq \overline{t_i}$ for all i. In other words, T_i contains all the $\Delta_i = \overline{t_i} - \underline{t_i} + 1$ integers from $\underline{t_i}$ to $\overline{t_i}$. When a duration value of an activity i is known with certainty, we have the deterministic case represented by an interval T_i where $\underline{t_i} = \overline{t_i}$.

According to [27, 49, 70, 71], we denote by $\mathcal{T}$ a *configuration* of the activity duration times in a given IIN S, and by $D_{\mathcal{T}}$ the related realization of a specific integer value from the interval activity time $[\underline{t_i}, \overline{t_i}]$ for $i = 1, \ldots, m$, whereas the dependence of the makespan from the configuration $\mathcal{T}$ is indicated with $C_{max}(\mathcal{T})$.

Hereinafter —by extension— we refer to $\mathcal{T}$ as a configuration of the network, and we indicate with $\mathrm{U}_{\mathcal{T}}(S)$ the set of all the configurations of an IIN S, which are obtained through the *Cartesian product* of the integer intervals associated with the durations of the activities [27, 49, 70, 71]. The total number of all possible configurations $|\mathrm{U}_{\mathcal{T}}(S)|$ of a network S can be determined as follows:

$$|\mathrm{U}_{\mathcal{T}}(S)| = \prod_{i=1}^{m} \Delta_i. \tag{4}$$

The IINs have an intrinsically discrete nature, and for a given IIN S, the makespan is finite and bounded keeping values in $[\underline{C}_{max}, \overline{C}_{max}]$, where $\underline{C}_{max}$ ($\overline{C}_{max}$) refers to the configuration $\underline{\mathcal{T}}$ ($\overline{\mathcal{T}}$) of S in which all durations are at their minimum (maximum) value. The finite length $\Gamma = \overline{C}_{max} - \underline{C}_{max} + 1$ of the interval values for the makespan $\mathbf{C}_{max}$ represents a measure of the amount of uncertainty of the IIN. The quantiles and superquantiles of the makespan are also finite and bounded holding the following:

$$\underline{C}_{max} \leq q_\alpha(\mathbf{C}_{max}) \leq Q_\alpha(\mathbf{C}_{max}) \leq \overline{C}_{max}, \forall \alpha \in [0, 1] \tag{5}$$

Furthermore, some other simple bounds can be considered. Let $C_{max}(E(\mathbf{D}))$ be the deterministic counterpart of the expected makespan, i.e., the completion time obtained when each activity duration $\mathbf{D}_i$ assumes exactly its expected value $E(\mathbf{D}_i)$, it is well known (e.g., see [10, 29]) that:

$$C_{max}(E(\mathbf{D})) \leq E(\mathbf{C}_{max}) \leq Q_\alpha(\mathbf{C}_{max}), \forall \alpha \in [0, 1] \tag{6}$$

Considering a configuration $\mathcal{T}$ of a given IIN S, let $l_{\mathcal{T}}(i, j)$ be the length of the *longest path* (assuming that the length of a path is the sum of the durations of the arcs belonging to it) from node i to node j. With this notation, the makespan $C_{max}(\mathcal{T})$ can be computed as $l_{\mathcal{T}}(1, n)$. The earliest and the latest starting time of an event i can be computed as $l_{\mathcal{T}}(1, i)$, and $C_{max}(\mathcal{T}) - l_{\mathcal{T}}(i, n)$, respectively [24].

Table 1: Summary of notations used for IINs

Notation	Description
$G = (N, A)$	precedence DAG with N nodes and A arcs
$m = \|A\|$	number of arcs in G
$\mathbf{D} = (\mathbf{D}_1, \ldots, \mathbf{D}_m)$	vector of random durations associated to the arcs
$\mathbf{C}_{max}$	IIN makespan as random variable
$E(\mathbf{C}_{max})$	expected value of the makespan
$E(\mathbf{D}_i)$	expected value of the random duration $\mathbf{D}_i$, $i = 1, \ldots, m$
$C_{max}(E(\mathbf{D}))$	makespan for the configuration in which each activity duration assumes its expected value
$T_i = [\underline{t_i}, \overline{t_i}]$	interval activity integers times, $i = 1, \ldots, m$
$\underline{t_i}, \overline{t_i}$	integer values with $\underline{t_i} \geq 0$, and $\underline{t_i} \leq \overline{t_i}$, for all $i = 1, \ldots, m$
$\Delta_i = \overline{t_i} - \underline{t_i} + 1$	length of T_i, $i = 1, \ldots, m$
$\mathcal{T}$	a specific configuration of the activity duration times
$D_{\mathcal{T}}$	realization of an exact value from the interval activity time associated to the configuration $\mathcal{T}$
$C_{max}(\mathcal{T})$	makespan associated to the configuration $\mathcal{T}$
$U_{\mathcal{T}}(S)$	set of all the configurations of an IIN S
$\|U_{\mathcal{T}}(S)\|$	number of all the configurations of an IIN S
$\underline{\mathcal{T}}$	the configuration in which all the durations are at their minimum value
$\overline{\mathcal{T}}$	the configuration in which all the durations are at their maximum value
$\underline{C}_{max}$	makespan for the configuration $\underline{\mathcal{T}}$
$\overline{C}_{max}$	makespan for the configuration $\overline{\mathcal{T}}$
Γ	length of the interval $[\overline{C}_{max} - \underline{C}_{max}]$
α	probability level used for the quantile risk measures
$q_\alpha(\mathbf{C}_{max})$	quantile of the makespan at level α
$Q_\alpha(\mathbf{C}_{max})$	superquantile of the makespan at level α
$d_{\mathcal{T}}(i, j)$	duration of the activity $a = (i, j)$ in the configuration $\mathcal{T}$
$l_{\mathcal{T}}(i, j)$	length of the longest path from node i to node j, in the configuration $\mathcal{T}$
$s_{\mathcal{T}}(i, j)$	slack time of the activity $a = (i, j)$ in the configuration $\mathcal{T}$
G_ξ	ξ-*critical subgraph* of G, i.e., the graph containing all the activities $a \in A \| s_{\mathcal{T}}(a) \leq \xi$.

A common measure of criticality of an activity $a = (i, j)$ is its *slack* time $s_{\mathcal{T}}(a)$, which is the amount of time we can increase the activity duration $d_{\mathcal{T}}(i, j)$ without affecting $C_{max}(\mathcal{T})$, thus:

$$s_{\mathcal{T}}(a) = s_{\mathcal{T}}(i, j) = C_{max}(\mathcal{T}) - l_{\mathcal{T}}(1, i) - d_{\mathcal{T}}(i, j) - l_{\mathcal{T}}(j, n). \tag{7}$$

An activity is said to be *critical* if it is on a *critical path* and its slack time $s_{\mathcal{T}}(a)$ is zero [24]. Since all the activities have integer valued durations, all the slack times are also integers. The makespan and the slack of each activity can be efficiently computed in a DAG G solving the related *longest path problem* [7, 31, 36]. As a TAN can have more critical paths, the set of critical activities forms a subgraph of G, and hereinafter we refer to it as *critical subgraph*. We extend this concept introducing the *ξ-critical subgraph* G_ξ of G as the graph containing all the activities $a \in A | s_{\mathcal{T}}(a) \leq \xi$.

The concept of *ξ-critical subgraph* can be applied to a IIN $(G, \mathbf{D})$ by indicating as G_ξ the ξ-critical subgraph of G containing all the arcs $a = (i, j) \in A | s_{\overline{\mathcal{T}}}(a) \leq \xi$, where the slack $s_{\overline{\mathcal{T}}}(a)$ of the arc a refers to the worst (or pessimistic) configuration $\overline{\mathcal{T}}$ obtained setting all the durations to their maximum value. More formally:

$$s_{\overline{\mathcal{T}}}(a) = s_{\overline{\mathcal{T}}}(i, j) = \overline{C}_{max} - l_{\overline{\mathcal{T}}}(1, i) - d_{\overline{\mathcal{T}}}(i, j) - l_{\overline{\mathcal{T}}}(j, n) \tag{8}$$

in which $d_{\overline{\mathcal{T}}}(i, j)$ is the maximum duration of the activity (i, j).

Table 1 summarizes the parameters and the notation associated to a IIN $(G, \mathbf{D})$.

4. Computation of the Quantiles and Superquantiles of the Makespan in IINs

The definitions of the quantile and the superquantile suggest a straightforward exact order statistics procedure to calculate the $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ in IINs [10, 62, 69]. In fact, considering a IIN S, the analyst can sort the configurations in $\mathrm{U}_{\mathcal{T}}(S)$ in decreasing C_{max} order, and check the CT_α highest (i.e., worst) configurations, where CT_α stands for *counting target* for the given value of α:

$$CT_\alpha = \lceil (1 - \alpha) |\mathrm{U}_{\mathcal{T}}(S)| \rceil \tag{9}$$

and then determine the corresponding values of $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ using the set $W_{CT_\alpha}(S)$ of the worst CT_α configurations of S.

Unfortunately, these simple ordinal quantile and superquantile procedures can be directly used only for very small cases and usually they are not viable in practice due to the huge number of configurations IINs commonly admit. In the literature, *sampling* or *scenario* based techniques are widely used to produce statistical estimations of both $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ [62]. However these techniques are in general very time consuming to yield results with acceptable confidence levels, and give results based on statistical prediction intervals not always convenient for decision makers to use [61, 62]. These difficulties, together with the need for statistical validation of the results, have recently been overcome with the introduction in [49] of an approach based on the calculation of theoretical lower and upper bounds which form an interval certainly (i.e., by construction) containing the exact value of the adopted risk indicator and which appears to be very tight in practice. Thanks to this theoretical certification of the obtained bounds, this approach can be also a useful tool to validate or guide other sample or scenario based methods.

Although the seminal approach proposed in [49] offers good results, applications (e.g., see [21, 50, 51]) may require better performance on both computation time and solution quality, especially in real-time applications. Therefore, in this paper we improve the counting based approach introduced in [49] for the makespan of a IIN, and extend it by proposing and testing a set of new algorithms to compute both $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$.

Given a IIN S represented by $(G, \mathbf{D})$, and the probability level $\alpha \in [0, 1]$, as the definition of $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ involves the set $W_{CT_\alpha}(S)$ of the worst (i.e., pessimistic) CT_α configurations in $\mathrm{U}_{\mathcal{T}}(S)$, our counting approach starts from the worst case (i.e., the configuration $\overline{\mathcal{T}}$) and proceeds backwards up to the CT_α-th worst configuration indicated as w_α.

Thus, the quantile $\mathrm{q}_\alpha(\mathbf{C}_{max})$ corresponds to the makespan of the configuration w_α, i.e., $q_\alpha(\mathbf{C}_{max}) = C_{max}(w_\alpha)$, while the superquantile $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ is the mean over the set $W_{CT_\alpha}(S)$:

$$Q_\alpha(\mathbf{C}_{max}) = \sum_{i \in W_{CT_\alpha}(S)} \frac{C_{max}(\mathcal{T}_i)}{CT_\alpha}. \tag{10}$$

Initially, the counting procedure individuates $\overline{\mathcal{T}}$, calculates $\overline{C}_{max}$, determines $|\mathrm{U}_{\mathcal{T}}(S)|$, and computes the counting target CT_α.

To start the counting process, the number of configurations $|\mathrm{U}_{\overline{\mathcal{T}}}(S)|$ having the worst makespan value $\overline{C}_{max}$ can be obtained, referring to a specific (integer) level of slack ξ, partitioning G in two subgraphs G_ξ and $G \setminus G_\xi$ as described at the end of Section 3.

Then $|\mathrm{U}_{\overline{\mathcal{T}}}(S)|$ can be calculated as the product of two terms $F_{1\xi}$ and $F_{2\xi}$. The first refers to the arcs in $G \setminus G_\xi$ and can be determined as

$$F_{1\xi} = \prod_{i \in G \setminus G_\xi} \Delta_i \tag{11}$$

In fact, all the arcs belonging in $G \setminus G_\xi$ have a slack value (i.e., a criticality) greater of ξ and all their possible configurations (i.e., durations) do not affect $\overline{C}_{max}$.

The second factor $F_{2\xi}$ represents the number of configurations having makespan $\overline{C}_{max}$ in G_ξ and requires a specific counting function $count(\overline{C}_{max}, G_\xi)$ to be evaluated (which will be illustrated in Section 4.3). Thus, the following holds:

$$|\mathrm{U}_{\overline{\mathcal{T}}}(S)| = F_{2\xi}F_{1\xi} = count(\overline{C}_{max}, G_\xi) \cdot \prod_{i \in G \setminus G_\xi} \Delta_i \tag{12}$$

Similarly, let $L \in (0, \xi)$ be an integer, we call $F_{2\xi L} = count(\overline{C}_{max} - L, G_\xi)$ the number of configurations in the graph G_ξ having makespan $\overline{C}_{max} - L$. Thus the product $F_{2\xi L}F_{1\xi}$ provides the overall number of configurations in G having makespan $\overline{C}_{max} - L$.

These concepts are the basis of the proposed algorithmic scheme.

4.1. General algorithmic scheme

This subsection describes the general algorithmic scheme adopted for the set of procedures we propose to calculate $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$ for a given IIN and a certain probability level α. The general algorithmic scheme is illustrated in

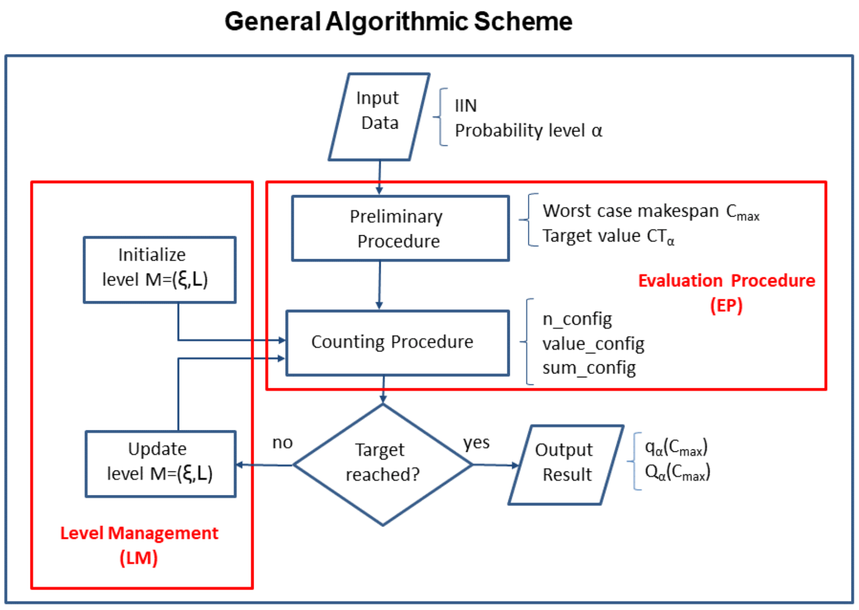

Figure 1: General algorithmic scheme for the computation of the quantiles and superquantiles of the makespan in IINs

Figure 1. It receives as input a IIN S represented by $(G, \mathbf{D})$ and the probability level α, and returns the corresponding evaluation of $\mathrm{q}_\alpha(\mathbf{C}_{max})$ and $\mathrm{Q}_\alpha(\mathbf{C}_{max})$. The main blocks of this scheme are the *Evaluation Procedure* (EP) and the *Level Management* (LM).

The EP block contains a Preliminary Procedure to determine the total number of possible configurations $|\mathrm{U}_{\mathcal{T}}(S)|$, the target number of configurations $CT\alpha$, and the worst or pessimistic value $\overline{C}_{max}$ from where to start the evaluation in the Counting Procedure. implementing the functions $F_{1\xi}$ and $F_{2\xi L}$. The first can be easily computed once G_ξ is known, whereas $F_{2\xi L}$ requires to be carefully designed.

The successive Counting Procedure uses the internal variables *n_config*, *sum_config*, and *value_config* to register the overall number of configurations, the cumulative sum of the related makespan, and the last makespan value considered in the numerical iterative process. The computation at each iteration refers to a specific level $M = (\xi, L)$ provided by the LM block, and proceeds backwards considering the configurations leading to each possible makespan value for that level. The overall Evaluation Procedure is implemented as an iterative process which continues until sufficient information has been gathered to compute the risk indices under study using the values of the internal variables *n_config*, *sum_config* and *value_config*.

The LM block implements the strategy for managing the values of ξ and L employed in the Counting Procedure. More specifically LM initializes the level to start the process, and updates it at each iteration as illustrated in Algorithm 1 where a pseudo-code of the General Algorithmic Procedure is reported.

The described algorithmic scheme is quite general, and for both LM and EP different possible implementations can be considered leading to different configurations of the algorithm.

4.2. Level Management Methods

The Level Management block provides the level $M = (\xi, L)$ containing the integer values of ξ and L used by the Evaluation Procedure. Iteratively, LM fixes a value of ξ and generates a sequence of levels $M = (\xi, L)$ where L is updated by assuming all integer values from 0 to ξ. Then ξ is incremented by Δ_ξ and the procedure is repeated. The level $M = (\xi, L)$ is initialized with $\xi = 0$ and $L = 0$. As for the update mode of ξ we consider two cases on the basis of the determination of the increment Δ_ξ of the value of ξ: *step-by-step* (indicated as St) and *slack-by-slack* (denoted as Sl).

In the first case (St), adopted also in [49], the increment Δ_ξ is unitary. Thus the sequence of values of ξ simply follows the natural integers.

```
GENERAL ALGORITHMIC PROCEDURE
Data: IIN=$(G, \mathbf{D}), \alpha$
Result: $Q_\alpha(\mathbf{C}_{max})$, $q_\alpha(\mathbf{C}_{max})$
Initialization phase:
  determine $\overline{C}_{max}$, $CT_\alpha$;
  LM sets $M = (\xi = 0, L = 0)$;
  set $Q_\alpha = q_\alpha = 0$;
  set $n\_config = sum\_config = value\_config = n_L = \delta_L = s_L = 0$;
while $n\_config \leq CT_\alpha$ do
  | LM updates $M = (\xi, L)$
  | $n_L = min(n\_config + F_{2\xi L}F_{1\xi}, CT_\alpha)$;
  | $\delta_L = n_L - n\_config$;
  | $value\_config = \overline{C}_{max} - L$;
  | $s_L = \delta_L value\_config$;
  | $sum\_config = sum\_config + s_L$;
  | $n\_config = n_L$;
end
Output:
  $Q_\alpha = \frac{(sum\_config)}{CT_\alpha}$, $q_\alpha = value\_config$
```

Algorithm 1: Pseudo-code of the General Algorithmic Procedure

Whereas, in the second case (Sl) the ξ values follows the sequence (in increasing order) of the slack values $s_{\overline{\mathcal{T}}}(a)$ of the arcs a referring to the worst configuration $\overline{\mathcal{T}}$ obtained setting all the durations to their maximum value. Thus Δ_ξ used in each iteration is the one needed to bring ξ to the next slack value. Since some levels (values) are skipped this approach reduces the total number of iterations of the algorithm.

4.3. Evaluation Procedure Methods

The Evaluation Procedure (EP) has been described in the general scheme reported in Figure 1 and in the Algorithm 1. The $F_{2\xi L}$ counting function devoted to compute $count(\overline{C}_{max}-L, G_\xi)$ plays a fundamental role in EP, and this Section is mainly devoted to describe in details its implementation.

Starting this section we introduce some further notations useful to describe the implementation of the counting functions. For a given IIN, as described in Section 3, the durations of its m activities are only known as the equally possible integer values included in a given interval $T_i = [\underline{t_i}, \overline{t_i}]$ for $i = 1, \ldots, m$ which represents each possible realizations of the duration $\mathbf{D}_i$ for the activity i. We associate to each activity i (with $i = 1, \ldots, m$) a data structure $\mathbf{IV}_i = (\mathbf{D}_i, \mathbf{O}_i)$ storing the relevant information about the *interval values*, namely two vectors of length Δ_i containing the integer time duration values $\mathbf{D}_i$, and the number of possible configurations $\mathbf{O}_i$ for each time duration value in T_i.

The implementation of the function $F_{2\xi L}$ is based on the idea that in the graph G_ξ some useful transformations can be iteratively applied to reduce G_ξ to a single *final* arc (i.e., from node i to n) carrying all the necessary information $\mathbf{IV}_{final} = (\mathbf{D}_{final}, \mathbf{O}_{final})$. Therefore the function $F_{2\xi L}$ can directly drawn its counting result from $\mathbf{IV}_{final}$. In fact, the latter contains the number of possible configurations (i.e., in $\mathbf{O}_{final}$) for each possible makespan value associated to G_ξ reported in $\mathbf{D}_{final}$.

To this aim, we have considered different possible transformations leading to a set of configurations of the overall algorithm. They are divided into exact transformations indicated as *reduction* procedures and heuristic transformations denoted as *approximation* procedures. They are described in the following sections.

4.3.1. Reduction Procedures

Often in IINs there are activities planned to be executed in parallel or in series which possibly induce series-parallel sub-structures in G_ξ [7, 9], and for this reason the following two reduction transformations are firstly considered.

Series Reduction. Let $a_h = (i, j)$ and $a_k = (u, v)$ two activities of G_ξ. If these activities are associated to arcs in G_ξ connected in series [7], (i.e., $j = u$, and j has unitary both its in-degree $deg^-(j)$ and out-degree $deg^+(j)$), then they can be replaced by an equivalent single arc (i, v) associated to the corresponding series of activities named ser_{hk}. This transformation needs to associate to ser_{hk} the $\mathbf{IV}_{ser_{hk}} = (\mathbf{D}_{ser_{hk}}, \mathbf{O}_{ser_{hk}})$, in which $\mathbf{D}_{ser_{hk}}$ has an associated time interval $T_{ser_{hk}} = [\underline{t_{ser_{hk}}}, \overline{t_{ser_{hk}}}]$, where $\underline{t_{ser_{hk}}} = \underline{t_h} + \underline{t_k}$, and $\overline{t_{ser_{hk}}} = \overline{t_h} + \overline{t_k}$;

while $\mathbf{O}_{ser_{hk}}$ can be determined by the discrete convolution in which $\mathbf{O}_h$ and $\mathbf{O}_k$ are considered as data samples displaced in time according to $\mathbf{D}_h$ and $\mathbf{D}_k$, respectively [18]. We indicate, in terms of interval value structures, this series reduction operation as $\mathbf{IV}_{ser_{hk}} = (\mathbf{IV}_h * \mathbf{IV}_k)$.

Parallel Reduction. Let $a_h = (i, j)$ and $a_k = (u, v)$ two activities of G_ξ. In case the activities a_h and a_k are associated to arcs connected in parallel [7] in G_ξ, (i.e., $i = u$ and $j = v$), then they can be replaced by an equivalent single arc (i, j), representing the activity par_{hk}. This transformation needs to associate to par_{hk} the $\mathbf{IV}_{par_{hk}} = (\mathbf{D}_{par_{hk}}, \mathbf{O}_{par_{hk}})$ in which $\mathbf{D}_{par_{hk}}$ has an associated time interval $T_{par_{hk}} = [\underline{t_{par_{hk}}}, \overline{t_{par_{hk}}}]$, where $\underline{t_{par_{hk}}} = max(\underline{t_h}, \underline{t_k})$, and $\overline{t_{par_{hk}}} = max(\overline{t_h}, \overline{t_k})$; while $\mathbf{O}_{par_{hk}}$ contains, for each element $d_{par_{hk}} \in \mathbf{D}_{par_{hk}}$, the correspondent number of possible configurations. The latter is combinatorially determined by the overall number of possible configurations of the original durations $d_h \in \mathbf{D}_h$ and $d_k \in \mathbf{D}_k$ of activities a_h and a_k, such that $max(d_h, d_k) = d_{par_{hk}}$. Thus, we indicate, in terms of interval value structures, the parallel reduction operation as $\mathbf{IV}_{par_{hk}} = \mathcal{MAX}(\mathbf{IV}_h, \mathbf{IV}_k)$.

These two transformations can be repeatedly applied to try to completely reduce a given directed subgraph G_ξ to a single final arc $(1, n)$ described by $\mathbf{IV}_{final}$. In that case, $\mathbf{IV}_{final}$ is independent from the specific adopted transformation sequence. This tentative will be successful if G_ξ is a *series-parallel* (SP) graph [7], even if G is not necessarily SP. Therefore, if all G_ξ are SP digraphs, the function $F_{2\xi L}$ returns exact evaluations. These transformations lead to the definition of a reduction procedure [49] which we refer to as SP method. Given a network S represented by $(G, \mathbf{D})$, and $\alpha \in [0, 1]$, if all G_ξ involved in the $F_{2\xi L}$ calls are SP digraphs, the General Algorithmic Scheme equipped with the SP method returns the exact value of Q_α and q_α.

In general, in addition to the series and parallel transformations, other transformations can be applied in particular cases contributing to the exact reduction of a given directed subgraph G_ξ. They are described below.

Y-Reduction. Let v be a node of G_ξ having in-degree $deg^-(v) = 1$, and $a_h = (u, v)$ be the activity associated to the unique arc ending in v. Each successor i of the node v has associated an activity $a_k = (v, i)$ having its own interval value structure $\mathbf{IV}_k$. In case the activity a_h has a deterministic duration (i.e., $T_h = [\underline{t_h}, \overline{t_h}]$, where $\underline{t_h} = \overline{t_h}$), it has only one possible configuration. Then, a transformation can be applied involving a_h and the activities a_k for all successors i. This transformation consists of a contraction of the arc (u, v) obtained collapsing nodes u and v, removing arc (u, v), and replacing each arc (v, i) with an arc $(u, i) = ser_{hk}$ locally applying the series transformation with $\mathbf{IV}_{ser_{hk}} = (\mathbf{IV}_h * \mathbf{IV}_k)$. When v has a unique successor i, this reduction operation coincides with an ordinary series transformation, but in the other cases neither the series nor the parallel transformation can be directly applied. A similar transformation can be applied, changing as necessary, in the case of a node v of G_ξ having out-degree $deg^+(v) = 1$.

D-Reduction. Let v be a node of G_ξ having $deg^-(v) = p$ and $deg^+(v) = q$. Let $a_h = (h, v)$ with $h = 1, \ldots, p$, be the activities associated to the arcs ending in v. Each successor k of the node v has associated an activity $a_k = (v, k)$, with $k = 1, \ldots, q$. In case all activities a_h and a_k have a deterministic duration (i.e., $T_h = [\underline{t_h}, \overline{t_h}]$, where $\underline{t_h} = \overline{t_h}$, $T_k = [\underline{t_k}, \overline{t_k}]$, with $\underline{t_k} = \overline{t_k}$), both of them have only one possible configuration. Then, a transformation can be applied involving the node v and all the activities a_h and a_k. This transformation consists of the removal of node v and all arcs either starting or ending in it. Then a set of new arcs (h, k) (i.e., with h and k predecessor and successor of v, respectively) is added with arc $(h, k) = ser_{hk}$ locally applying the series transformation with $\mathbf{IV}_{ser_{hk}} = (\mathbf{IV}_h * \mathbf{IV}_k)$.

IG-Reduction. If previous exact reduction methods fails to complete their reduction process, they return a *residual* directed subgraph $G'_\xi \subseteq G_\xi$, instead of a single final arc. This means that G_ξ contains a subgraph homeomorphic to the so called *interdictive graph* [7, 9] depicted in Figure 2. In order to improve the reduction process of the graph G_ξ, possibly to a single final arc, we consider the application of a new reduction transformation called Interdictive Graph reduction, indicated as IG-Reduction for short. The transformation is

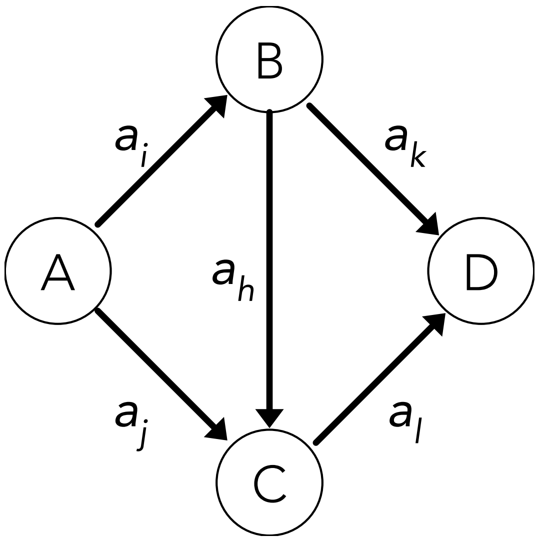

Figure 2: The interdictive graph

applied to interdictive graphs as represented in Figure 2 when the nodes B and C have no other entering or leaving edges in G_ξ (whereas, nodes A and D possibly have). This transformation returns an arc (A, D) associated to an activity IG_{ijhkl} with the $\mathbf{IV}_{IG_{ijhkl}}$ equivalent to the interdictive graph which is therefore replaced with it by simplifying the graph G_ξ. Alternatively considering node B (having $deg^-(B) = 1$) or node C (having $deg^+(C) = 1$) the procedure IG-Reduction obtains the $\mathbf{IV}_{IG_{ijhkl}}$ of the equivalent arc A, D applying the Y-Reduction and SP transformations previously introduced. Without loss of generality, we describe the application of the procedure to node B in Figure 2. We observe that if activity a_i has a deterministic duration the Y-Reduction can be applied obtaining two equivalent arcs (A, C) and (A, D) replacing activities a_i, a_h, and a_k. Then successive series-parallel transformations yield the final equivalent arc (A, D) and its $\mathbf{IV}_{IG_{ijhkl}}$. In general, a_i is not deterministic and has its own $\mathbf{IV}_i = (\mathbf{D}_i, \mathbf{O}_i)$. Nevertheless, for each value in $\mathbf{D}_i$ the reduction can be obtained as a case with a deterministic a_i activity. Then the overall equivalent $\mathbf{IV}_{IG_{ijhkl}} = (\mathbf{D}_{ijhkl}, \mathbf{O}_{ijhkl})$ is determined including all the configurations obtained in the performed reductions.

The extension of the SP method with the *Y-Reduction*, *D-Reduction*, and *IG-Reduction* transformations allows us to define a new and advanced reduction procedure, hereinafter indicated as *Adv* method. Given a network S represented by $(G, \mathbf{D})$, and $\alpha \in [0, 1]$, if all G_ξ involved in the $F_{2\xi L}$ calls are fully reduced by Adv method, the General Algorithmic Scheme returns exact values for Q_α and q_α.

Nevertheless, the transformation process based on either SP or Adv reduction methods applied to G_ξ can yield a (partially simplified) residual graph G'_ξ on which there are no other eligible arcs for further reductions. In this latter case, in order to reduce G_ξ to a single final arc, we have to apply also some different transformations of G'_ξ that, however, do not guarantee the exactness of the counts of $F_{2\xi L}$. However the use of these approximation procedures can produce useful lower and upper bounds of both Q_α and q_α.

4.3.2. Approximation procedures

When computing $F_{2\xi L}$, if the exact reduction methods fails to complete their reduction process, they return a *residual* directed subgraph $G'_\xi \subseteq G_\xi$, instead of a single final arc. This means that G_ξ contains some subgraph homeomorphic to the so called *interdictive graph* [7, 9]. It is worth noting that in general, even in the case of D-Reduction or IG-Reduction application, not all interdictive graphs can be reduced. In order to force the reduction of the residual digraph G'_ξ to a single final arc, we consider the application of the two following transformations.

Arc removal. This transformation removes from G'_ξ one arc $a_h = (i, j)$ with $\mathbf{IV}_h$ in order to destroy an interdictive subgraph structure possibly allowing the application of some transformation to further reduce the resulting graph. Thus, considering the structure of the interdictive graph, an arc (i,j) is eligible to be removed if it satisfies the following two conditions: *a)* for node i: $deg+(i) \geq 2$, $deg^{-}(i) \geq 1$; and *b)* for node j: $deg^{+}(j) \geq 1$, $deg^{-}(j) \geq 2$. The transformation consists in the cancellation of the arc a_h from the current subgraph G'_{ξ}.

Arc replication. This transformation replicates in G'_{ξ} one arc $a_h = (i,j)$ with $\mathbf{IV}_h$ so that the interdictive subgraph can be destroyed through the possible application of further transformations. Considering the structure of the interdictive graph, an arc (i,j) is eligible to be replicated if satisfies one of the following conditions: *a)* for node i: $deg^{-}(i) \geq 2$, $deg^{+}(i) = 1$; or *b)* for node j: $deg^{-}(j) = 1$, $deg^{+}(j) \geq 2$. The transformation replaces, in the current subgraph G'_{ξ}, the arc a_h with its R_h replicas (each having the same $\mathbf{IV}_h$). In case *a) arc replication* removes the original arc $a_h = (i,j)$ and introduces $R_h = deg^{-}(i)$ new arcs having a common extreme in node j, while the other extreme nodes are obtained *splitting* node i in R_h new nodes, each of which has exactly one incoming arc between those that had the original node i. Similarly, in case *b) arc replication* removes the original arc $a_h = (i,j)$ and introduces $R_h = deg^{+}(j)$ new arcs having a common extreme in node i, while the other extreme nodes are obtained *splitting* node j in R_h new nodes, each of which has exactly one outgoing arc between those that had the original node j.

These approximations contrast the presence of interdictive graphs that prevent the application of further reduction operations. We observe that an arc eligible for their application always exists in G'_{ξ} when it cannot be fully reduced. Thus, once one of these approximations has been applied, the counting function $F_{2\xi L}$ starts a new exact reduction phase. This process is (possibly) repeated, and terminates only when one single final arc is obtained. In general, the application of either arc removal or arc replication can introduce an approximation in the final arc data $\mathbf{IV}_{final}$ thus affecting the results of $F_{2\xi L}$. Note that, the arc removal transformation never adds configurations to those corresponding to the initial subgraph G'_ξ, this induces the overall procedure to produce under-estimations in $\mathbf{IV}_{final}$. On the other hand, the arc replication transformation never eliminates configurations to those corresponding to G'_ξ, this induces the overall procedure to produce over-estimations in $\mathbf{IV}_{final}$.

These transformations can be exploited to obtain lower and upper bounds of the overall evaluation of Q_α and q_α according to the following Propositions 1 and 2.

Proposition 1. *Given a network S represented by $(G, \boldsymbol{D})$, and $\alpha \in [0,1]$, if the graphs G_ξ involved in the $F_{2\xi L}$ calls are not all SP digraphs, and only SP or Adv reductions and arc removal transformations are applied to obtain the $\boldsymbol{IV}_{final}$ representations, the General Algorithmic Scheme returns a lower bound LB_{Q_α} [LB_{q_α}] of the overall evaluation of Q_α [q_α].*

Proof. Recall that Q_α and q_α are estimated by means of a counting procedure to identify the set $W_{CT_\alpha}(S)$ of the worst CT_α configurations in $\mathrm{U}_{\mathcal{T}}(S)$. Given an initial subgraph G'_ξ, the arc removal transformation never adds configurations to the set of the worst configurations $W_{CT_\alpha}(S)$. On the other hand, the arc removal transformation may eliminate configurations from $W_{CT_\alpha}(S)$ which will be replaced in the counting procedure by configurations having a smaller C_{max}, thus leading to under-estimations in $\mathbf{IV}_{final}$ and a lower bound estimation of both Q_α and q_α. $\square$

Proposition 2. *Given a network S represented by $(G, \boldsymbol{D})$, and $\alpha \in [0,1]$, if the graphs G_ξ involved in the $F_{2\xi L}$ calls are not all SP digraphs, and only SP or Adv reductions and arc replication transformations are applied to obtain the $\boldsymbol{IV}_{final}$ representations, the General Algorithmic Scheme returns an upper bound UB_{Q_α} [UB_{q_α}] of the overall evaluation of Q_α [q_α].*

Proof. Recall that Q_α and q_α are estimated by means of a counting procedure to identify the set $W_{CT_\alpha}(S)$ of the worst CT_α configurations in $\mathrm{U}_{\mathcal{T}}(S)$. Given an initial subgraph G'_ξ, the arc replication transformation never removes configurations from the set of the worst configurations $W_{CT_\alpha}(S)$. On the other hand, the arc replication transformation may add configurations with higher C_{max} to $W_{CT_\alpha}(S)$ thus causing over-estimations in $\mathbf{IV}_{final}$ and an upper bound estimation of both Q_α and q_α. $\square$

If the procedure devoted to determine the final arc $\mathbf{IV}_{final}$ associated to G_ξ, uses only arc removal (arc replication) transformations in addition to exact reduction methods SP or Adv, then $F_{2\xi L}$ always under-estimates (over-estimates) its counts, inducing the overall algorithm to yield a lower (upper) bound of Q_α and q_α. However, they can coincide when, due to the particular structure of the graphs G_ξ, the structural perturbations introduced by the applied transformations do not lead to alterations of the counts.

According to the bounds considered in Equations (5) and (6), determining a lower bound of Q_α based on the arc removal approximation, the algorithm gives in output the maximum between $C_{max}(E(\mathbf{D}))$ and the obtained counting result. In some cases, the number of cancelled configurations may lead to an insufficient configurations in the final arc data with respect to the counting target CT_α. Then, the algorithm associates to the missing configurations the value $\underline{C}_{max}$ in order to complete its computation ensuring to get a lower bound.

Both the considered approximation methods apply a transformation on an arc to be selected in a set of eligible candidates. In our implementation we consider two variants for these approximation methods. The differences between them lie in the management of the set of eligible arcs for removal and replication operations. The following two cases are considered: mS and mS2.

mS. The procedure finds all the eligible arcs in G'_ξ and, in both LB and UB cases, selects arc a_e with the minimum sum S of elements in $\mathbf{O}_e$. The rationale being to introduce smaller perturbations in the graph structures and therefore smaller approximations in the counts.

mS2. The procedure finds all the eligible arcs in G'_ξ. Then, in the LB case selects arc a_e with the minimum sum S of elements in $\mathbf{O}_e$ belonging to the critical path. Whereas in the UB case it selects arc a_e with the minimum sum S of elements in $\mathbf{O}_e$ multiplied by the replicas of the arc introduced by the procedure. This setting represents a choice based on a more accurate evaluation of the perturbation introduced by the approximations.

4.4. Algorithmic configurations to determine Q_α *and* q_α *of the makespan*

We adopt the General Algorithmic Scheme introduced in Section 4.1 and consider all possible configurations for the methods dedicated to the level management, reduction and approximation transformations introduced in Sections 4.2 and 4.3.

More specifically, eight configurations can be obtained combining the following:

- *level management methods*: the step-based method (St), and the slack-based method (Sl);
- *reduction methods*: the basic Series-Parallel (SP) method and the Advanced method (Adv);
- *approximation methods*: the basic method mS and its variant mS2.

Each of these configurations produces both a lower bound LB_α and an upper bound UB_α for Q_α and q_α of the makespan which can be estimated as:

$$\widehat{Q_\alpha} = \frac{(UB_{Q_\alpha} + LB_{Q_\alpha})}{2}. \tag{13}$$

$$\widehat{q_\alpha} = \frac{(UB_{q_\alpha} + LB_{q_\alpha})}{2}. \tag{14}$$

When the obtained upper and lower bounds have the same value, the result is exact, and we say that the algorithmic configuration has *closed* the instance.

Regarding the computational complexity of the proposed algorithmic approach we observe what follows.

Firstly, we observe that the SP transformations are pseudo-polynomial since they depend on the interval values of arcs (i.e., **IV**). In particular, their complexity depend on the amount of uncertainty of the IIN represented by the length Γ (in terms of number of integers) of the time interval $[\underline{C}_{max}, \overline{C}_{max}]$. In fact, discrete convolution (series) and $\mathcal{MAX}$ (parallel) operations are executed in $O(\Gamma^2)$ and $O(\Gamma)$, respectively. Regarding Y, D and IG transformations, they require at most $O(n)$, $O(n^2)$ and $O(\Gamma)$ series operations, respectively.

If the graph is fully reducible to a single arc by a sequence of reduction transformations, the whole reduction process requires a number of transformations which is linear [7, 9] with respect to the size of the graph (i.e., in the number of arcs m). Furthermore, if the graph is irreducible the number of necessary approximations transformations (i.e. arc removal and arc replication) is at most $O(m)$. Therefore when the SP reduction is used, the different configurations of this algorithm are all $O(\Gamma^2 m^2)$, whereas they are $O(\Gamma^3 m^2)$ adopting the Adv method.

Each algorithm configuration is indicated with a three-field coding in which the first field contains the description of the level management method, the second indicates the type of reduction and the third the approximation method. The state-of-the-art algorithm presented in [49] corresponds to the configuration St/SP/mS, and is assumed as benchmark in this study, while all the others present innovative aspects that are the subject of the experimental investigation.

All the eight algorithmic configurations have been implemented and tested in a wide experimental campaign reported in the following Section.

4.5. Illustrative example

In this Section we discuss a small example to illustrate the proposed approach. For a given IIN, as described in Section 3, the durations of its m activities are only known as the equally possible integer values included in a given interval $T_i = [\underline{t_i}, \overline{t_i}]$ for $i = 1, \ldots, m$ which represents each possible realizations of the duration $\mathbf{D}_i$ for the activity i.

Figure 3 shows a given IIN S for which we are interested in computing

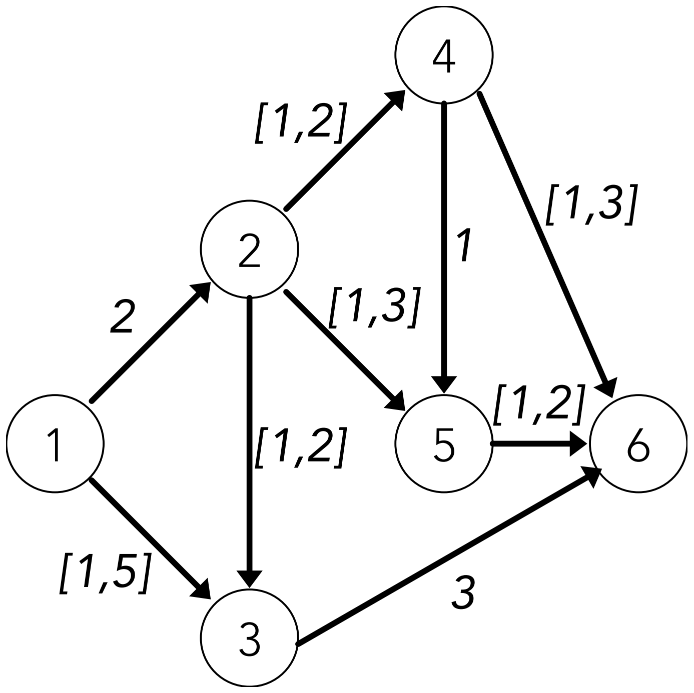

Figure 3: Illustrative example: IIN S.

$q_{\alpha}(\mathbf{C}_{max})$ and $Q_{\alpha}(\mathbf{C}_{max})$ risk measures at the probability level $\alpha = 0.5$ using the the St/SP/mS and St/Adv/mS algorithmic configurations.

The given network S has $m = 9$, and shows $\overline{C}_{max} = 8$, $\underline{C}_{max} = 6$ giving $\Gamma = 3$, whereas the total number of configurations can be calculated using Equation (4) as $|\mathrm{U}_{\mathcal{T}}(S)| = 360$. Hence, in order to obtain the desired q_{α} and Q_{α} of the makespan, the counting procedure will need to count a target obtained applying Equation (9) as $CT_{\alpha} = 180$ configurations.

In the first iteration, the counting procedure starts from the level $M = (\xi, L) = (0, 0)$ associated with the worst makespan $\overline{C}_{max} = 8$. At this level the critical graph G_{ξ} is composed by two arcs, namely $(1, 3)$ and $(3, 6)$. It is clearly a series-parallel graph, and $F_{2\xi}$ admits only one configuration with makespan value 8. The overall configurations associated to this level are then $F_{2\xi}F_{1\xi} = 72$, which being lower than the target CT_{α} requires the procedure to continue to the next level $\xi = 1$ associated with $C_{max} = 7$.

In the new iteration, we can observe that all the arcs of the original graph are critical, and the resulting critical graph G_ξ is not series-parallel. Hence, the SP procedure cannot successfully reduce that graph to a single arc. The algorithm St/SP/mS presented in [49] invokes the approximation transformation mS to obtain the lower and upper bounds, counting a total of 69 and 288 configurations, respectively. Thus the upper bound on $Q_\alpha(\mathbf{C}_{max})$ can be computed as 7.4, while the lower bound still does not count a sufficient number of configurations and the algorithm needs to explore the successive level $\xi = 2$ associated with $C_{max} = 6$ in which the counting target CT_α is reached. At the end of the run the St/SP/mS algorithm on the basis of the obtained bounds returns for $Q_{0.5}(\mathbf{C}_{max})$ the interval $[7.183, 7.400]$, while for $q_{0.5}(\mathbf{C}_{max})$ it returns the interval $[6, 7]$. In this case the obtained solution is not exact, and the final results can be calculated averaging the bounds yielding $Q_\alpha(\widehat{\mathbf{C}_{max}}) = 7.292$, and $q_\alpha(\widehat{\mathbf{C}_{max}}) = 6.5$, respectively. In this case the algorithmic configuration is not able to close the instance.

On the other hand, using the St/Adv/mS algorithmic configuration, Adv reduction procedure, in addition to SP transformations, can further apply Y and IG transformations counting exactly a total of 225 configurations already at the level $\xi = 1$ associated with $C_{max} = 7$. Hence, in this case $Q_{0.5}(\mathbf{C}_{max}) = 7.4$, and $q_{0.5}(\mathbf{C}_{max}) = 7$, and in both cases the obtained solution is exact since lower and upper bound have the same value, closing the instance. Thus, this algorithmic configuration obtained a more accurate result by analyzing a smaller number of levels.

5. Experimental Study

This experimental analysis is devoted to test eight algorithmic configurations considering the procedures proposed in Sections 4-4.3. These configurations are obtained combining the following:

- *level management methods*: St and Sl
- *reduction methods*: SP and Adv

- *approximation methods*: mS and mS2

Each algorithm configuration is indicated with a three-field coding as introduced in Section 4.4.

All the numerical experiments were conducted on a Intel Core i9-9920X processor (4.4 GHz clock speed) operating under Linux with 32 GB of RAM. The code has been written in Python 3.10 and run on a single thread.

5.1. Test instances

In this computational study four test-sets have been generated starting by four PSPLIB's sets: J30, J60, J90, and J120 [32, 37, 38]. PSPLIB is a widely used instance library for resource-constrained project scheduling problems, considered to be representative of real scheduling problems. The considered sets contain instances having 30, 60, 90, and 120 activities, respectively. Each set includes 480 cases, except J120 which has 600 instances. Therefore, the considered test-sets contain a total of 2040 instances.

To obtain our test-sets, we adopt the same procedure proposed in [49] to systematically modify the original PSPLIB's instances to: *(i)* remove resources limitations, *(ii)* obtain AoA networks and *(iii)* introduce uncertainties.

A preliminary analysis on the networks of all these cases shows that no instance is associated to a simple series-parallel digraph. Since the overall amount of uncertainty (i.e., in terms of Γ) may have a relevant impact on the computational effort required to evaluate the considered quantile-based measures, the rationale for this design of experiments is to get a larger and more challenging set of test instances than the ones used in [49]. Therefore, we consider different parameters. Regarding the relevance and the level of uncertainties in the networks we adopt the parameter ρ representing the percentage of the number of uncertain activities and we consider $\rho \in \{0.25, 0.5, 1\}$. The generation of the uncertain duration $\mathbf{D}_a$ for an activity a is based on the value of its deterministic duration d_a in the original PSPLIB's instance. This value is used as reference to randomly generate the time interval $T_a = [\underline{t_a}, \overline{t_a}]$ as the only necessary information associated with $\mathbf{D}_a$. In fact, as described in Section

3, the activity durations are only known as the equally possible integer values included in a given interval for each activity a. To this aim we assume the first extreme of the interval as $\underline{t_a} = d_a$, then we generate the other extreme as $\overline{t_a} = random(U[\lceil(1 + 0.5\gamma)d_a\rceil, \lceil(1 + \gamma)d_a\rceil])$, where $0 \leq \gamma \leq 1$ is a parameter for which we consider the values in the set $\{0.25, 0.5, 1\}$. The settings obtained combining these parameters provide nine scenarios characterized by different degrees of uncertainty. Summing up, considering the original PSPLIB's instances and all the settings (i.e., for ρ and γ), the whole number of instances included in our experimental campaign is 18360. Note that, the larger instance in terms of Γ admits 6.4×10^{62} configurations. Moreover, we consider three different values for the probability level α of the quantile-based measures, namely: $\alpha \in \{0.90, 0.95, 0.99\}$. This triples the total number of analyzed cases to 55080.

5.2. Results

We have tested the eight algorithms proposed in Section 4 on the test-sets described in Section 5.1. We have run all the algorithms on each of the 55080 risk evaluations requiring in the worst case a maximum computation time of 12.27 s.

Table 2: Computational results: general overview of evaluations (in %).

Sets	**Closed**	**Open**	**SP-closed**	**Adv-closed**	**H-closed**
J30	96.75	3.25	87.14	95.42	1.33
J60	92.69	7.31	79.21	90.89	1.80
J90	90.04	9.96	75.73	88.24	1.80
J120	86.77	13.23	72.33	84.62	2.15
ALL	91.28	8.72	78.23	89.49	1.79

A first aggregate analysis of the obtained results refers to the whole data set and is summarized in Table 2 which reports a general overview of evaluations (in %). The first column reports the sets of instances, while the successive two columns report the percentage of *Closed* and *Open* instances, respectively. After the computational experiments, an instance is *Closed* if it is possible to determine the exact value of $q_\alpha(\mathbf{C}_{max})$ and $Q_\alpha(\mathbf{C}_{max})$, i.e., for at least one algorithm, the corresponding lower and upper bounds coincide. Otherwise the instance is considered *Open* as these bounds define an interval of possible values of the risk index of interest. In these cases, only heuristic estimations of the quantile-based measures are available, and they are obtained averaging the values of the bounds. This situation is due to the presence in the IIN of structures that the algorithms are unable to simplify such as general subgraphs homeomorphic to the interdictive graph. Then, successive approximation procedures can introduce changes of such importance as to preclude the possibility of obtaining an exact computation. On the other hand, when this perturbation is slight and the configurations to be counted do not change, exact results can still be achieved. The results reported highlight that larger instances are more difficult to solve exactly. Nevertheless, the overall results reported in the last row indicate that for 91.28% of the evaluations exact results were obtained.

In the other columns, we report for each group of instances the percentages of instances exactly solved using the SP and Adv reduction method (indicated as *SP-closed* and *Adv-closed*); and, finally, the percentage of further instances closed by the heuristic approach (*H-closed*).

Recalling that even if no graph is series-parallel (as discussed in Section 5.1) in the test sets, the considered G_ξ may be, we call *SP-closed* instances those (78.23% in total) for which the algorithms applying only series-parallel reductions on G_ξ (regardless of whether G is a series-parallel graph) are able to reach their counting target CT_α. In other words, regardless of whether the whole network G is a series-parallel graph, all the critical graph structures G_ξ processed by the algorithms are series-parallel. Similarly, *Adv-closed* cases are those solved exactly by algorithms that use the Adv reduction method. They achieve the 89.49% and include the *SP-closed* cases. The remaining 10.51% of cases is not completely solved using Adv reductions, and calls for the application of an approximation method to completely reduce the graph to a single arc as discussed in Section 4.4. These approximations are performed by heuristic reduction methods able to close the instances in the 17.05% of cases they are used, and 1.79% of the overall instances were closed only after the use of the heuristics.

We complete the general overview of the aggregated results considering the behavior of the considered eight algorithm configurations. This analysis is summarized in Tables 3 and 4 for algorithms adopting the Step-based and the Slack-based level management, respectively.

The first row in these tables reports the percentage of instances closed by each algorithm (in columns).

Table 3: General overview of results for algorithms using the Step-based level management.

Index	**St/Adv/mS**	**St/Adv/mS2**	**St/SP/mS**	**St/SP/mS2**
%Closed	91.22	90.96	82.15	81.54
I_Q**% (avg.)**	2.68	2.82	6.43	6.71
I_Q**% (st.d.)**	12.01	12.33	18.65	18.97
MRE_Q**% (avg.)**	0.26	0.27	0.58	0.6
MRE_Q**% (st.d.)**	1.22	1.24	1.77	1.8
I_q**% (avg.)**	5.45	5.75	12.41	12.95
I_q**% (st.d.)**	23.7	24.42	35.05	35.7
MRE_q**% (mean)**	0.57	0.6	1.19	1.23
MRE_q**% (st.d.)**	2.61	2.67	3.64	3.7
Time (mean)	0.11	0.11	0.1	0.11
Time (st.d)	0.56	0.58	0.4	0.41

Table 4: General overview of results for algorithms using the Slack-based level management.

Index	**Sl/Adv/mS**	**Sl/Adv/mS2**	**Sl/SP/mS**	**Sl/SP/mS2**
%Closed	91.22	91.1	82.15	81.85
I_Q**% (avg.)**	2.68	2.77	6.43	6.61
I_Q**% (st.d.)**	12.01	12.25	18.65	18.88
MRE_Q**% (avg.)**	0.26	0.27	0.58	0.59
MRE_Q**% (st.d.)**	1.22	1.24	1.77	1.79
I_q**% (avg.)**	5.45	5.66	12.41	12.77
I_q**% (st.d.)**	23.7	24.23	35.05	35.51
MRE_q**% (mean)**	0.57	0.59	1.19	1.22
MRE_q**% (st.d.)**	2.61	2.65	3.64	3.68
Time (mean)	0.08	0.08	0.07	0.08
Time (st.d)	0.47	0.49	0.33	0.34

Each configuration of the algorithm is able to close a high percentage (ranging from 81.54% to 91.22%) of instances showing in general a slight better performance when the Slack-based level management is used. Moreover, the configurations adopting the Adv reduction method and mS approximation perform better than others with everything else being equal.

The successive four rows report the mean and the standard deviation of two indicators we adopt to assess the quality of the results obtained for Q_α averaged on the instances. The first indicator is the *quality index* I_Q which is defined as follows:

$$I_Q = \begin{cases} 0 & \text{if} \quad \overline{C}_{max} - C_{max}(E(\mathbf{D})) = 0 \\ \frac{UB_{Q_\alpha} - LB_{Q_\alpha}}{\overline{C}_{max} - C_{max}(E(\mathbf{D}))} & \text{otherwise} \end{cases} \tag{15}$$

I_Q compares the gap between the bounds (UB_{Q_α} and LB_{Q_α}) on Q_α obtained by the algorithm and the distance between the worst case and the deterministic counterpart of the makespan (which are considered as the *initial* knowledge on those bounds). In other words, I_Q represents a measure of the relative reduction of the uncertainty on the knowledge of Q_α obtained by the application of the algorithms.

The second indicator is the *estimated maximum relative error* MRE_Q which is determined considering that the approximation adopted when the algorithm is not able to find the exact evaluation, can cause an absolute error of at most $\frac{UB_{Q_\alpha}(\mathbf{C}_{max}) - LB_{Q_\alpha}(\mathbf{C}_{max})}{2}$, and therefore guarantees a maximum relative error MRE_Q on the specific solution evaluation quantified as follows:

$$MRE_Q = \frac{UB_{Q_\alpha}(\mathbf{C}_{max}) - LB_{Q_\alpha}(\mathbf{C}_{max})}{UB_{Q_\alpha}(\mathbf{C}_{max}) + LB_{Q_\alpha}(\mathbf{C}_{max})} \tag{16}$$

Both indicators reveal a good overall behavior of the algorithms which, on average, present 4.64% and 0.43% for I_Q and MRE_Q, respectively. Tables 3 and 4 report also their standard deviations for each configuration. These quality measures on Q_α show a better behavior of St/Adv/mS and Sl/Adv/mS, while highlight St/SP/mS2 as the worst performing algorithm on average.

Similarly, the following block of four rows in Tables 3 and 4 report the mean and the standard deviation of the two indicators adopted to assess the quality of the results for q_α. The I_q and MRE_q indicators are conceptually similar to those previously described but are adapted to the case of q$_\alpha$ considering the related bounds and a different reference interval:

$$I_q = \begin{cases} 0 & \text{if} \quad \overline{C}_{max} - \underline{C}_{max}) = 0 \\ \frac{UB_{q_\alpha} - LB_{q_\alpha}}{\overline{C}_{max} - \underline{C}_{max})} & \text{otherwise} \end{cases} \tag{17}$$

$$MRE_q = \frac{UB_{q_\alpha}(\mathbf{C}_{max}) - LB_{q_\alpha}(\mathbf{C}_{max})}{UB_{q_\alpha}(\mathbf{C}_{max}) + LB_{q_\alpha}(\mathbf{C}_{max})} \tag{18}$$

The relative behavior of the algorithms substantially confirms what was observed for Q$_\alpha$.

The last rows of Tables 3 and 4 show the computation times (in seconds) of each algorithm, averaged on all the instances and the related standard deviations. The algorithms required on average 0.093 seconds, and configurations using the Slack-based level management result faster. More specifically Sl/SP/mS is the fastest configuration but it is not the best in terms of quality of the solutions for both the quantile-based measures.

We can observe that on the average the state-of-the-art algorithm proposed in [49] turns out to be outperformed by all configurations adopting the Adv reduction method. Among these, those that use slack-based level management are the fastest, while the best performing configuration on average, considering both results quality and computation time, is Sl/Adv/mS.

Table 5: Methods Comparison for Q_α over all the experimental cases

Index	**Methods Comparison**					
I_Q	St	0.51	97.72	1.78	Sl	Level management
	SP	0.07	82.55	17.38	Adv	Reduction
	mS	4.44	93.13	2.43	mS2	Approximation
MRE_Q	St	0.51	97.72	1.78	Sl	Level management
	SP	0.07	82.55	17.38	Adv	Reduction
	mS	4.45	93.13	2.43	mS2	Approximation
Time	St	28.47	0.00	71.53	Sl	Level management
	SP	6.88	0.00	93.12	Adv	Reduction
	mS	49.27	0.14	50.58	mS2	Approximation

To better analyze the contribute of each single method both to the quality

Table 6: Methods Comparison for q_α over all the experimental cases

Index	**Methods Comparison**					
I_q	St	0.30	98.54	1.15	Sl	Level management
	SP	0.07	83.55	16.38	Adv	Reduction
	mS	3.17	94.92	1.91	mS2	Approximation
MRE_q	St	0.30	98.54	1.15	Sl	Level management
	SP	0.07	83.55	16.38	Adv	Reduction
	mS	3.20	94.89	1.91	mS2	Approximation
Time	St	28.47	0.00	71.53	Sl	Level management
	SP	6.88	0.00	93.12	Adv	Reduction
	mS	49.27	0.14	50.58	mS2	Approximation

Table 7: Methods Comparison for Q_α over the open cases

Index	**Methods Comparison**					
I_Q	St	3.12	86.00	10.88	Sl	Level management
	SP	0.59	7.31	92.10	Adv	Reduction
	mS	29.78	49.92	20.30	mS2	Approximation
MRE_Q	St	3.12	86.00	10.88	Sl	Level management
	SP	0.59	7.31	92.10	Adv	Reduction
	mS	29.83	49.92	20.25	mS2	Approximation
Time	St	4.71	0.00	95.29	Sl	Level management
	SP	53.29	0.00	46.71	Adv	Reduction
	mS	57.48	0.02	42.50	mS2	Approximation

Table 8: Methods Comparison for q_α over the open cases

Index	**Methods Comparison**					
I_q	St	1.78	91.39	6.83	Sl	Level management
	SP	0.62	15.93	83.46	Adv	Reduction
	mS	20.90	62.85	16.25	mS2	Approximation
MRE_q	St	1.78	91.39	6.83	Sl	Level management
	SP	0.64	15.90	83.47	Adv	Reduction
	mS	21.12	62.65	16.23	mS2	Approximation
Time	St	4.71	0.00	95.29	Sl	Level management
	SP	53.29	0.00	46.71	Adv	Reduction
	mS	57.48	0.02	42.50	mS2	Approximation

and the efficiency of the overall procedure Tables 5 and 6 report the comparison conducted for Q_α and q_α, respectively. For each performance index (i.e., quality, relative error, and computation time) all pairs of methods for Level management (St vs Sl), Reduction (SP vs Adv), and Approximation (mS vs mS2) are compared over all the 55080 computational experiments when the remaining algorithm's configuration is equal. For all comparisons the tables report the percentage of cases each method performs better (by rows, in the first and the third numerical entry, respectively), while in the central column is reported the percentage of cases in which the compared methods offer the same level of performance.

On the basis of these results we can observe in general that no method is fully outperformed by its competitor. In fact, there are cases in which each method is preferable.

Concerning both quality indicators (I_Q and I_q) and relative errors (MRE_Q and MRE_q), in most of the cases the behavior of the compared methods is similar. Nevertheless, Adv achieves a percentage of success clearly higher than SP. The differences among the other compared methods are more limited, in particular in the q_α risk measure.

The comparison on the computation times shows better results for the new methods introduced in this paper (i.e., Sl, Adv, and mS2), however, also in this case the general observation holds as no method performs better in all the cases.

A similar comparative analysis is conducted on the set of *open* cases, and the results are summarized in Tables 7 and 8. For these more difficult instances to solve, a greater difference is observed in the behavior and contribution of the different methods. Among the Reduction methods, Adv remains, in percentage terms, by far the best. The successful behavior of Sl in terms of Level management is less marked. As for the Approximation methods, mS and mS2 both offer a valid contribution, in many cases being complementary in finding better solutions, and in about 50% of cases they offer solutions with the same quality. As regards the computation times, only on Level management there is a marked better behavior of Sl with respect to St. In the other two cases, a substantially complementary behavior of the compared methods is observed.

The results obtained show that the proposed methods for Q_α and q_α of the makespan of IINs have a very good behavior in terms of quality of solutions and measured relative errors. We also observe that in the overall behaviors of the algorithm configurations there is a correlation between the average results for the two indicators. This can be explained by the fact that configurations that perform comparatively well on average under one performance indicator tend to perform well also on the other measure.

The analysis of these results shows that the proposed procedures allows the method to significantly improve the quality of the risk assessment with respect to the state-of-the art algorithm (corresponding to the configuration St/SP/mS). While they reduce the required computation time, thus improving over the state-of-the-art algorithm. More specifically, while the prevalence of Sl for the Level management and Adv for the reduction technique is quite marked, the prevalence of mS2 as approximation method is rather slight.

6. Conclusions

This paper deals with scheduling problems represented by temporal networks with incomplete and uncertain activity durations. More precisely, it considers the case of Integer Interval-Valued Activity Networks where durations can assume, as in different pratical contexts, only equally possible integer values belonging to a known interval.
Given a IIN representing a feasible solution for a scheduling problem, we developed a set of methods to be used in a counting algorithmic scheme to evaluate the quantile and the superquantile of the makespan as risk indicators. The evaluation gives as result theoretically proven lower and upper bounds.

The counting procedure, starting from the pessimistic makespan (i.e., the worst possible makespan), counts backwards how many configurations lead to each considered makespan value. This counting process is iterated until sufficient information has been gathered to compute the quantile based risk measures at a desired probability level α.

The computation at each step is done by exploiting the contribution of three methods devoted to level management, graph reduction and counting approximation, respectively. For each method two different procedures are considered and tested in an extensive experimental campaign carried out on the basis of classic project scheduling benchmarks.

More precisely, these methods allow us to configure eight overall algorithms extending the state-of-the-art and providing fast computation of the quantile and superquantile indicators yielding extremely good results. They highlight an algorithm with great possibilities of application in real-time or massive and repetitive scheduling activities.

The proposed approach can enable to exploit the benefits and effectiveness of using the quantile based risk indicators as optimization criteria for a wide classes of scheduling problems considering the risk attitude of the scheduler in different practical contexts.
Given the short computation times requirements and the offered quality of the evaluations, one possible research direction is to embed this approach in an optimization algorithm. In optimization algorithmic schemes the proposed evaluation of the risk could be done at every step of an iterative algorithm (i.e., for example in a local search) or it could be carried out only on selected solutions for a validation or ex-post assessment of the risk associated to a given solution.

Different, yet interesting, research directions to consider, include possible extensions of the proposed approach. They could aim at developing further algorithmic improvements or more effective exact or heuristic schemes; considering more general uncertainty representations such as general probability functions and fuzzy sets [21] also aimed to a distributionally robust analysis; applying or adapting the proposed approach to the case of different risk indicators or utility functions.

References

[1] Allahverdi A, Aydilek H, Aydilek A (2014) Single machine scheduling problem with interval processing times to minimize mean weighted completion time. *Computers & Operations Research*, 51, 200-207.

[2] Artigues C, Leus R, Talla Nobibon F (2013) Robust optimization for

resource-constrained project scheduling with uncertain activity durations. *Flexible Services and Manufacturing Journal*, 25, 175–205.

[3] Atakan S, Bülbül K, Noyan N (2017) Minimizing value-at-risk in single-machine scheduling. *Annals of Operations Research* 248, (1–2), 25–73.

[4] Aydilek A, Aydilek H, Allahverdi A (2013) Increasing the profitability and competitiveness in a production environment with random and bounded setup times. *International Journal of Production Research*, 51, 106-117.

[5] Baker KR (2014) Setting optimal due dates in a basic safe–scheduling model *Computers & Operations Research* 41, 109–114.

[6] Baker K, Trietsch D (2009) *Principles of sequencing and scheduling.* John Wiley & Sons, New York.

[7] Bang-Jensen J, Gutin G (2008) *Digraphs: Theory, Algorithms and Applications,* 2nd edition. Springer-Verlag, London.

[8] Batur D, Choobineh F (2010) A quantile-based approach to system selection, *European Journal of Operational Research*, 202, 3, 764–772.

[9] Bein WM, Kamburowski J, Stallmann MFM (1992) Optimal Reduction of Two-Terminal Directed Acyclic Graphs. *SIAM Journal on Computing*, 21, 6, 1112–1129.

[10] Bertsimas D, Lauprete GJ, Samarov A (2004) Shortfall as a risk measure: properties, optimization and applications. *Journal of Economic Dynamics & Control* 28, (7), 1353–1382.

[11] Brucker P, Knust S (2013) *Complex Scheduling.* (Springer), 2nd edition.

[12] Bruni ME, Guerriero F, Pinto E (2009) Evaluating project completion time in project networks with discrete random activity durations. *Computers & Operations Research*, 36 (10), 2716–2722.

[13] Canon LC, Jeannot E (2010) Evaluation and Optimization of the Robustness of DAG Schedules in Heterogeneous Environments. *IEEE Transactions on Parallel and Distriduted Systems* 21, (4), 532–546.

[14] Catalão JPS, Pousinho HMI, Contreras J (2012) Optimal hydro scheduling and offering strategies considering price uncertainty and risk management. *Energy*, 37, 237–244.

[15] Chanas S, Zieliński P (2002) The computational complexity of the critical problems in a network with interval activity times. *European Journal of Operational Research* 136, 541–550.

[16] Chanas S, Dubois D, Zieliński P (2002) On the Sure Criticality of Tasks in Activity Networks With Imprecise Durations. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics* 32, (4) 393–407.

[17] Chang Z, Song S, Zhang Y, Ding J-Y, Zhang R, Chiong R (2017) Distributionally robust single machine scheduling with risk aversion. *European Journal of Operational Research* 256, 261–274.

[18] Damelin S, Miller W (2011) *The Mathematics of Signal Processing*, Cambridge University Press.

[19] Daniels RL, Kouvelis P (1995) Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science* 41 (2), 363-376.

[20] De P, Ghosh JB, Wells CE (1992) Expectation-variance analyss of job sequences under processing time uncertainty. *International Journal of Production Economics* 28(3), 289–297.

[21] Dellino G, Meloni C, Pranzo M, Samà M (2022) Expected Shortfall for the Makespan in Activity Networks with Fuzzy Durations, *IEEE Transactions on Fuzzy Systems*, 30(6), 2124–2128.

[22] Demeulemeester EL, Herroelen WS (2002) *Project scheduling a research handbook*. Kluwer Academic, Dordrecht.

[23] De Reyck B, Demeulemeester E, Herroelen W (1999) Algorithms for Scheduling Projects with Generalized Precedence Relations. In: Węglarz J. (eds) *Project Scheduling*. International Series in Operations Research and Management Science, vol 14. Springer, Boston.

[24] Elmaghraby SE (1977) *Activity Networks: Project Planning and Control by Network Models*. Wiley, New York.

[25] Elmaghraby SE (1990) Project bidding under deterministic and probabilistic activity durations. *European Journal of Operational Research* 49, 14–34.

[26] Elmaghraby SE (2005) On the fallacy of averages in project risk management. *European Journal of Operational Research* 165, (2), 307–313.

[27] Fortin J, Zieliński P, Dubois D, Fargier H (2010) Criticality analysis of activity networks under interval uncertainty. *Journal of Scheduling*, 13, 609-627.

[28] Framinan JM, Leisten R, Ruiz García R (2014) *Manufacturing Scheduling Systems. An Integrated View on Models, Methods and Tools.* Springer-Verlag, London.

[29] Fulkerson DR (1962) Expected critical path lengths in PERT networks. *Operations Research* 10, 808–817.

[30] García-González J, Parrilla E, Mateo A (2007) Risk-averse profit-based optimal scheduling of a hydro-chain in the day-ahead electricity market. *European Journal of Operational Research*, 181, 1354–1369,

[31] Hagstrom JN (1988) Computational Complexity of PERT Problems. *Networks* 18, (2), 139–147.

[32] Hartmann S, Kolisch R (2000) Experimental evaluation of state-of-the-art heuristics for resource constrained project scheduling. *European Journal of Operational Research* 127, (2), 394–407.

[33] Herroelen WS, Leus R (2005) Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, 165, (2), 289–306.

[34] Ivanescu CV, Fransoo JC, Bertrand JWM (2002). Makespan estimation and order acceptance in batch process industries when processing times are uncertain. *OR Spectrum* 24, 467–495.

[35] Kalinchenko K, Veremyev A, Boginski V, Jeffcoat DE, Uryasev S (2011) Robust connectivity issues in dynamic sensor networks for area surveillance under uncertainty. *Pacific Journal of Optimization* 7, 235–248.

[36] Kelley JE Jr (1961) Critical-Path Planning and Scheduling: Mathematical Basis. *Operations Research* 9, (3), 296–320.

[37] Kolisch R, Hartmann S (2006) Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research* 174, (1), 23–37.

[38] Kolisch R, Sprecher A (1996) PSPLIB - A project scheduling library. *European Journal of Operational Research* 96, 205–216.

[39] Lai T-C, Sotskov YN (1999) Sequencing with uncertain numerical data for makespan minimisation *Journal of the Operational Research Society* 50, 230–243.

[40] Larsen R, Pranzo M (2019) A framework for dynamic rescheduling problems *International Journal of Production Research*, 57, 1, 16–33.

[41] Lawrence SR, Sewell EC (1997) Heuristic, optimal, static, and dynamic schedules when processing times are uncertain *Journal of the Operations Management* 15, 71–82.

[42] Li Z, Ierapetritou M (2008) Process scheduling under uncertainty: Review and challenges. *Computers and Chemical Engineering* 32, 715–727.

[43] Liao L, Sarin SC, Sherali HD (2012) A scenario generation-based lower bounding approach for stochastic scheduling problems, *Journal of the Operational Research Society*, 63, 1410-1420.

[44] Lin X, Janak SL, Floudas CA (2004) A new robust optimization approach for scheduling under uncertainty: I. Bounded uncertainty. *Computers & Chemical Engineering* 28, (6-7), 1069–1085.

[45] Liu R, Xie X, Yu K, Hu Q (2018) A survey on simulation optimization for the manufacturing system operation, *International Journal of Modelling and Simulation*, 38, 2, 116–127.

[46] Luh PB, Chen D, Thakur LS (1999) An Effective Approach for Job-Shop Scheduling with Uncertain Processing Requirements. *IEEE Transactions on Robotics and Automation* 15, 2, 715–727.

[47] Makridakis S, Wheelwright SC, Hyndman RJ (2008) Forecasting Methods and Applications. *3rd ed. Wiley.*

[48] Meloni C, Pacciarelli D, Pranzo M (2004) A Rollout Metaheuristic for Job Shop Scheduling Problems. *Annals of Operations Research* 131, (1-4), 215–235.

[49] Meloni C, Pranzo M (2020) Expected shortfall for the makespan in activity networks under imperfect information. *Flexible Services and Manufacturing Journal*, 32, 668–692.

[50] Meloni C, Pranzo M, Samà M (2021) Risk of delay evaluation in real-time train scheduling with uncertain dwell times. *Transportation Research Part E: Logistics and Transportation Review*, 152, 102366.

[51] Meloni C, Pranzo M, Samà M (2022) Evaluation of VaR and CVaR for the makespan in interval valued blocking job shops. *International Journal of Production Economics*, 247, 108455.

[52] Montemanni R (2007) A Mixed Integer Programming Formulation for the Total Flow Time Single Machine Robust Scheduling Problem with Interval Data. *Journal of Mathematical Modelling and Algorithms*, 6, 287-296.

[53] Pinedo M (1983) Stochastic scheduling with release dates and due dates. *Operations Research* 31, 559–72.

[54] Pinedo M (2001) *Scheduling: Theory, algorithms, and systems.* (Prentice Hall, Upper Saddle, NJ), 2nd edition.

[55] Pranzo, M, Meloni C, Pacciarelli D (2003) A New Class of Greedy Heuristics for Job Shop Scheduling Problems. *Lecture Notes in Computer Science* 2647, 223–236. Springer.

[56] Pranzo M, Pacciarelli D (2016) An iterated greedy metaheuristic for the blocking job shop scheduling problem. *Journal of Heuristics* 22 (4), 587–611.

[57] Ramponi FA, Campi MC (2017) Expected shortfall: Heuristics and certificates. *European Journal of Operational Research*, 267, 3, 1003–1013.

[58] Rockafeller RT (2007) Coherent approaches to risk in optimization under uncertainty. *INFORMS Tutorials in Operations Research*, 38–61.

[59] Rockafeller RT, Royset JO (2013) Superquantiles and Their Applications to Risk, Random Variables, and Regression. *INFORMS Tutorials in Operations Research*, 151–167.

[60] Sang P, Begen MA, Cao J (2021) Appointment scheduling with a quantile objective problems. *Computers & Operations Research* 132, 105295.

[61] Sarin, SC, Nagarajan B, Liao L (2010) *Stochastic scheduling: Expectation-variance analysis of a schedule*. Cambridge University Press, New York.

[62] Sarin SC, Sherali HD, Liao L (2014) Minimizing conditional-value-at-risk for stochastic scheduling problems. *Journal of Scheduling* 17, 5–15.

[63] Sterna M (2021) Late and early work scheduling: A survey. *Omega*, 104 102453.

[64] Tao L, Wu DD, Liu S, Dolgui A (2018) Optimal due date quoting for a risk–averse decision-maker under CVaR. *International Journal of Production Research*, 56, 5, 1934–1959.

[65] Urgo M, Váncza J (2019) A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. *Flexible Services and Manufacturing Journal*, 31, 472–496.

[66] Vieira GE, Herrmann JW, Lin E (2003) Rescheduling manufacturing systems: A framework of strategies, policies and methods. *Journal of Scheduling*, 6, 39–62.

[67] Wang J (2002) A fuzzy project scheduling approach to minimize schedule risk for product development. *Fuzzy Sets and Systems*, 127, 99-116.

[68] Wang J (2004) A fuzzy robust scheduling approach for product development projects. *European Journal of Operational Research*, 152, 180-194.

[69] Wiesemann W (2012) *Optimization of Temporal Networks Under Uncertainty*. Springer, Heidelberg.

[70] Yakhchali SH, Ghodsypour SH (2010) Computing latest starting times of activities in interval-valued networks with minimal time lags. *European Journal of Operational Research*, 200, 874-880.

[71] Zieliński P (2005) On computing the latest starting times and floats of activities in a network with imprecise durations. *Fuzzy Sets and Systems*, 150, 53-76.