Interval analysis in scheduling

Jérôme Fortin[1], Paweł Zieliński[2], Didier Dubois[1], and Hélène Fargier[1]

[1] IRIT/UPS 118 route de Narbonne, 31062, Toulouse, cedex 4, France,
{fortin, dubois, fargier}@irit.fr
[2] Institute of Mathematics, Wrocław University of Technology, Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland,
pziel@im.pwr.wroc.pl

Abstract. This paper reconsiders the most basic scheduling problem, that of minimizing the makespan of a partially ordered set of activities, in the context of incomplete knowledge. While this problem is very easy in the deterministic case, its counterpart when durations are interval-valued is much trickier, as standard results and algorithms no longer apply. After positioning this paper in the scope of temporal networks under uncertainty, we provide a complete solution to the problem of finding the latest starting times and floats of activities, and of locating surely critical ones, as they are often isolated. The minimal float problem is NP-hard while the maximal float problem is polynomial. New complexity results and efficient algorithms are provided for the interval-valued makespan minimization problem.

1 Introduction and Motivation

Temporal Constraint Networks (TCN) represent relations between dates of events and also allow to express constraints on the possible durations of activities from intervals of values [1]. To ensure a solution to the assignment problem, it is sufficient to check the well-known consistency properties of the network.

TCN have been extended to take into account uncertainty of durations of some tasks in realistic applications. A distinction is made between so-called contingent constraints (for example, when the duration of a task cannot be known before the execution of the task) and controllable ones (for example a time interval to be chosen between starting times of two tasks). The resulting network (called Simple Temporal Network with Uncertainty or STPU) becomes a decision-making problem under uncertainty, and notions of consistency must be refined so as to ensure controllability, that is, ensured consistency despite uncertainty [2–5]. As far as we know, the TCN community has extensively worked on the controllability of a network, but the question of optimizing the total duration of set of tasks described by a STPU has not been studied. Nevertheless, not all solutions to an STPU are equally valuable, and solutions minimizing the makespan are of obvious practical interest.

Given a set of tasks and a set of precedence constraints between them, the most elementary scheduling problem is to find the time window for the starting

time of each task in order to ensure a minimal overall duration or makespan. When all task durations are precise, the well-known PERT/CPM (Critical Path Method) algorithm provides such time-windows for all tasks in polynomial time (see [6]). In particular, a subset of tasks is found to be critical (their starting time-windows reduce to a singleton), and they form a set of critical paths. When the duration of tasks is ill-known and modeled by intervals, this problem can be viewed as a special kind of STPU where all tasks are modeled by contingent constraints, and controllable constraints only describe precedence between tasks. Of course, the resulting network is always controllable if the graph of precedence constraints is acyclic. However, the problem of minimizing the makespan in the interval-valued setting is much more difficult. It seems to have received attention only recently [7]. This concern actually derives from the literature on fuzzy PERT, a topic existing since the early eighties (see [8] for a survey). Especially Buckley [9] was the first to propose a rigorous formulation for this problem. Recent results show that in the presence of uncertainty the usual backward recursion algorithm for finding latest starting times of tasks are no longer applicable and the usual critical path analysis totally fails [10]. Yet, in the scope of selecting solutions to STPU which minimize the makespan, it is a basic problem to be solved in a first step.

Instead of being critical or not, tasks now form three groups: those that are for sure critical despite uncertainty (necessarily critical tasks), those that are for sure not critical, and tasks whose criticality is unknown, called possibly critical tasks (see [11]). Necessarily critical paths may fail to exist while necessarily critical tasks may be isolated. Finding the 3-partition of tasks in an interval-valued network turns out to be a NP-hard problem. Preliminary complexity results have recently appeared in [7], but some problems like the complexity of proving the necessary criticality of a task remained open. This paper provides a full picture of the complexity of the makespan minimization problem under the representation of interval-based uncertainty, and a set of efficient algorithms for determining the criticality of tasks, the optimal intervals containing their least ending times and their floats. It is shown that the only NP-hard problem is the one of finding the greatest lower bound of the float, which is closely related to asserting the possible criticality of a task. All other problems turn out to be polynomial. The fact that the two problems of asserting if an arc is necessarily critical or possibly critical do not have the same complexity is rather unexpected.

2 Preliminaries

An activity network is classically defined as a set of activities (or tasks) with given duration times, related to each other by means of precedence constraints. When there are no resource constraints, it can be represented by a directed, connected and acyclic graph. Of major concern, is to minimize the ending time of the last task, also called the makespan of the network. For each task, three quantities have practical importance for the management of the activity network: The *earliest starting time* est_{ij} of an activity (i, j) is the date before which the

activity cannot be started without violation of a precedence constraint. The *latest starting time* lst_{ij} of an activity (i,j) is the date after which the activity cannot be started without delaying the end of the project. The *float* f_{ij} of an activity (i,j) is the difference between the latest starting time lst_{ij} and the earliest starting time est_{ij}. An activity is *critical* if and only if its float is equal to zero. Under the assumption of minimal makespan, critical tasks must be started and completed at prescribed time-points.

3 The interval-valued scheduling problem

A directed, connected and acyclic graph $G =< V, A >$, represents an activity network. We use the activity-on-arc convention. V is the set of nodes (events), $|V| = n$, and A is the set of arcs (activities), $|A| = m$.

The set $V = \{1, 2, \ldots, n\}$ is labeled in such a way that $i < j$ for each activity $(i,j) \in A$. Activity durations (weights of the arcs) $(i,j) \in A$ are only known to belong to time intervals $D_{ij} = [d_{ij}^-, d_{ij}^+]$, $d_{ij}^- \geq 0$. Two nodes 1 and n are distinguished as the initial and final node, respectively.

We introduce some additional notations.

- A configuration is a precise instantiation of the duration of all tasks $(i,j) \in A$. Ω denotes a configuration, while $d_{ij}(\Omega) \in D_{ij}$ denotes the duration of activity (i,j) in configuration Ω.
  Let $B \subseteq A$ be a subset of activities. The configuration Ω_B^+ such that
  $d_{ij}(\Omega_B^+) = \begin{cases} d_{ij}^+ & \text{if } (i,j) \in B \\ d_{ij}^- & \text{otherwise} \end{cases}$ is called an *extreme configuration induced by* B.
- $\mathfrak{C}$ is the set of possible configurations of activity durations, $\mathfrak{C} = \times_{(i,j)\in A} D_{ij}$.
- $P(u,v)$ is the set of all paths $p(u,v)$ in G from node u to node v, we denote by P the set of all paths p in G from node 1 to node n.
- $l_p(\Omega)$ denotes the length of a path $p \in P(u,v)$ in Ω, $l_p(\Omega) = \sum_{(i,j)\in p} d_{ij}(\Omega)$.
- $Succ(i)$ (resp. $Pred(i)$) refers to the set of nodes that immediately follow (resp. precede) node $i \in V$.
- $SUCC(i,j)$ (resp. $PRED(i,j)$) denotes the set of all arcs that come after (resp. before) $(i,j) \in A$.
- $SUCC(i)$ (resp. $PRED(i)$) stands for the set of all nodes that come after (resp. before) $i \in V$.
- $G(i,j)$ is the subgraph of G composed of nodes succeeding i and preceding j.
- $G(D_{ij} = d)$ is the graph where duration of task (i,j) is replaced by d.

Computing earliest starting dates is not a difficult issue. Here we solve four problems , originally stated in [11, 12].

The first one is that of determining the widest intervals LST_{kl} (bounds) of possible values of the latest starting times lst_{kl} of a given activity $(k,l) \in A$, i.e. the interval $LST_{kl} = [lst_{kl}^-, lst_{kl}^+]$ defined by $lst_{kl}^- = \min_\Omega lst_{kl}(\Omega)$ and $lst_{kl}^+ = \max_\Omega lst_{kl}(\Omega)$. $lst_{kl}(\Omega)$ is the float of activity (k,l) in configuration Ω. The problem of computing the greatest lower (resp. least upper) bound of the latest

starting times is denoted GLBLST (resp. LUBLST). The second problem is that of determining the widest intervals F_{kl} of possible values of floats (total floats) f_{kl} of a given activity $(k,l) \in A$, i.e. the interval $F_{kl} = [f_{kl}^-, f_{kl}^+]$ bounded by $f_{kl}^- = \min_\Omega f_{kl}(\Omega)$ and $f_{kl}^+ = \max_\Omega f_{kl}(\Omega)$. $f_{kl}(\Omega)$ is the float of activity (k,l) in configuration Ω. The problem of computing the greatest lower (resp. least upper) bound of the floats is denoted GLBF (resp. LUBF). In both problems minimization and maximization are taken over all possible configurations $\mathfrak{C}$.

The next two problems are closely related to the ones defined previously. That is the problem of deciding the *possible criticality* of an activity and the problem of deciding the *necessary criticality* of an activity. An activity $(k,l) \in A$ is *possibly critical* in G if and only if there exists a configuration $\Omega \in \mathfrak{C}$ such that (k,l) is critical in G in Ω. An activity $(k,l) \in A$ is *necessarily critical* in G if and only if for every configuration $\Omega \in \mathfrak{C}$, (k,l) is critical in G.

There are obvious connections between the notions of criticality and the bounds on the float of an activity.

Proposition 1 *An activity $(k,l) \in A$ is possibly (resp. necessarily) critical in G if and only if $f_{kl}^- = 0$ (resp. $f_{kl}^+ = 0$).*

The solutions to problems GLBLST, LUBLST, LUBF and GLBF come down to finding an extreme configuration [11] where such bounds are attained. As there are 2^n extreme configurations, it explains the potentially exponential nature of the problem. GLBLST, LUBLST and GLBF have recently been solved in [7, 13], and only GLBF is NP-hard. In this paper, we recall the solutions of these three problems, and present the solution to the last one LUBF, thus providing a full picture of the makespan minimization problem under incomplete information.

4 Computational methods for evaluating criticality

This section presents a new method which can decide if a given task (k,l) is necessarily critical. First, under the assumption that the durations of the predecessors of task (k,l) are precisely known, we recall algorithms that respectively assert if (k,l) is possibly and necessarily critical. They constitute the basis for computing the LUB and GLB of the latest starting dates in polynomial time in [13]. We extend these results and give a general algorithm which asserts if (k,l) is necessarily critical in a network G in polynomial time without any consideration of the durations of tasks preceding (k,l). This result will lead in the next section to a polynomial algorithm which computes the LUB of the float of an activity.

Let us recall characteristic conditions of the non-necessary criticality of tasks.

Lemma 1 ([8]) *An activity $(k,l) \in A$ is not necessarily critical in G if and only if there exists a path $p \in P$ such that $(k,l) \notin p$, p is critical in configuration Ω_p^+ and no critical path in Ω_p^+ includes (k,l).*

Observation 1 *An activity $(k,l) \in A$ is not necessarily critical in G if and only if (k,l) is not critical in an extreme configuration in which the duration of (k,l) is at its lower bound and all activities from set $A \backslash SUCC(k,l) \backslash PRED(k,l) \backslash \{(k,l)\}$ have durations at their upper bounds.*

Now under the assumption that activities preceding (k,l) have precise durations, we can set the durations of tasks succeeding (k,l) at precise values while maintaining the status of (k,l) in terms of necessary criticality. It yields a configuration where (k,l) is critical if and only if it is necessarily critical in the interval-valued network. These durations are given by Propositions 2 and 3.

Proposition 2 *Let $(k,l) \in A$ be a distinguished activity, and (i,j) be an activity such that $(i,j) \in SUCC(k,l)$. Assume that every activity $(u,v) \in PRED(i,j)$ has precise duration. If (k,l) is critical in $G(1,i)$, then the following conditions are equivalent:*

(i) (k,l) is necessarily critical in G,
(ii) (k,l) is necessarily critical in $G(D_{ij} = d_{ij}^-)$.

Proof. $(i) \Longrightarrow (ii)$ Obvious.
$(i) \Longleftarrow (ii)$ We use a proof by contraposition. We need to prove that if (k,l) is critical in $G(1,i)$ and (k,l) is not necessarily critical in G, then (k,l) is not necessarily critical in $G(D_{ij} = d_{ij}^-)$. By assumption, (k,l) is not necessarily critical in G. From Lemma 1, it follows that there exists a path $p \in P$ such that $(k,l) \notin p$, p is critical in configuration Ω_p^+ and no critical path in Ω_p^+ includes (k,l). Since (k,l) is critical in $G(1,i)$, $(i,j) \notin p$ in Ω_p^+. Observe that $d_{ij}(\Omega_p^+) = d_{ij}^-$. From this and the fact that (k,l) is not critical in Ω_p^+, we conclude that (k,l) is not necessarily critical in $G(D_{ij} = d_{ij}^-)$. □

Proposition 3 *Let $(k,l) \in A$ be a distinguished activity, and (i,j) be an activity such that $(i,j) \in SUCC(k,l)$. Assume that every activity $(u,v) \in PRED(i,j)$ has precise duration. If (k,l) is not critical in $G(1,i)$, then the following conditions are equivalent:*

(i) (k,l) is necessarily critical in G,
(ii) (k,l) is necessarily critical in $G(D_{ij} = d_{ij}^+)$.

Proof. $(i) \Longrightarrow (ii)$ Straightforward.
$(i) \Longleftarrow (ii)$ To prove that (k,l) is necessarily critical in G, we only need to show that (k,l) is necessarily critical in $G(D_{ij} = d_{ij}^-)$. By assumption, (k,l) is necessarily critical in $G(D_{ij} = d_{ij}^+)$. From this, it follows that for every configuration in which the duration of (i,j) is at its upper bound, there exists a critical path traversing (k,l). Moreover, this path does not use (i,j), since (k,l) is not critical in $G(1,i)$. Thus the criticality of (k,l) does not depend on the duration of (i,j). Hence, (k,l) is critical for each configuration in which the duration of (i,j) is at its lower bound and consequently (k,l) is necessarily critical in $G(D_{ij} = d_{ij}^-)$. The necessary criticality of (k,l) in $G(D_{ij} = d_{ij}^-)$ and in $G(D_{ij} = d_{ij}^+)$ implies the necessary criticality of (k,l) in G. □

Propositions 2 and 3, together with Observation 1, lead us to Algorithm 1 for asserting the necessary criticality of a given activity (k,l) in a network in which all activities that precede (k,l) have precise durations. Testing if (k,l) is critical in $G(1,i)$ can be done in constant time because we already know if (k,l) is critical in $G(1,j)$ for all $j \in Pred(i)$, and so Algorithm 1 runs in $O(m)$.

We now present an algorithm for evaluating the necessary criticality of a fixed activity $(k,l) \in A$ in network G with interval durations, without any restriction. The key to the algorithm lies in Propositions 4 and 5 that enable a network with interval durations to be replaced by another network with precise durations for

Algorithm 1:

Input: A network G, activity (k,l), interval durations D_{uv}, $(u,v) \in A$ and for every task in $PRED(k,l)$ the duration is precisely given.
Output: `true` if (k,l) is necessarily critical in G; and `false` otherwise.
foreach $(u,v) \notin SUCC(k,l)$ **do** $D_{uv} \leftarrow d^+_{uv}$; $D_{kl} \leftarrow d^-_{kl}$;
for $i \leftarrow l$ **to** $n-1$ **such that** $i \in SUCC(l) \cup \{l\}$ **do**
  if (k,l) *is critical in* $G(1,i)$ **then foreach** $j \in Succ(i)$ **do** $D_{ij} \leftarrow d^-_{ij}$;
  else foreach $j \in Succ(i)$ **do** $D_{ij} \leftarrow d^+_{ij}$;
end
if (k,l) *is critical in* $G(1,n)$ **then return** `true`;
else return `false`;

activities preceding a fixed (k,l), in such a way that (k,l) is necessarily critical in the former if and only if it is necessarily critical in the latter.

Proposition 4 *Let $(k,l) \in A$ be a distinguished activity, and (i,j) be an activity such that $(i,j) \in PRED(k,l)$. If (k,l) is necessarily critical in $G(j,n)$, then the following conditions are equivalent:*
(i) (k,l) is necessarily critical in G,
(ii) (k,l) is necessarily critical in $G(D_{ij} = d^-_{ij})$.

Proof. The proof goes in the similar manner to the one of Proposition 2. □

Proposition 5 *Let $(k,l) \in A$ be a distinguished activity, and (i,j) be an activity such that $(i,j) \in PRED(k,l)$. If (k,l) is not necessarily critical in $G(j,n)$, then the following conditions are equivalent:*
(i) (k,l) is necessarily critical in G,
(ii) (k,l) is necessarily critical in $G(D_{ij} = d^+_{ij})$.

Proof. $(i) \Longrightarrow (ii)$ The proof is immediate.
$(i) \Longleftarrow (ii)$ We need to show that if (k,l) is not necessarily critical in $G(j,n)$ and (k,l) is necessarily critical in $G(D_{ij} = d^+_{ij})$, then (k,l) is necessarily critical in G. To prove this, assume on the contrary that (k,l) is not necessarily critical in G. From Lemma 1, it follows that there exists a path $p \in P$ such that $(k,l) \notin p$, p is critical in configuration Ω^+_p and no critical path in Ω^+_p includes (k,l) or equivalently (k,l) is not critical in Ω^+_p. We will show that for each such configuration, where (k,l) is not critical, the other assumptions lead to construct a critical path that traverses (k,l), which results in a contradictions. By assumption, (k,l) is not necessarily critical in $G(j,n)$. Then there exists a path $\hat{p} \in P(j,n)$ such that $(k,l) \notin \hat{p}$, $\hat{p}$ is critical in configuration $\Omega^+_{\hat{p}}$ and no critical path of $G(j,n)$ traverses (k,l) in $\Omega^+_{\hat{p}}$ (see Lemma 1).

Consider the extreme configuration induced by $p \cup \hat{p} \cup \{(i,j)\}$ and denote it by Ω^*. Note that (k,l) is critical in Ω^*, since (k,l) is necessarily critical in $G(D_{ij} = d^+_{ij})$. Thus there exists a critical path $p^* \in P$ using (k,l) in Ω^*. We define r to be the common node of $p^*(1,k)$ and p such that $r = \arg\max\{v \mid v \in V, v \in p^*(1,k), v \in p\}$ and define $\hat{r}$ to be the common node of $p^*(1,k)$ and $\hat{p}$ such that $\hat{r} = \arg\max\{v \mid v \in V, v \in p^*(1,k), v \in \hat{p}\}$.

We claim that if node $\hat{r}$ exists, then $\hat{r} = r$ or node $\hat{r}$ lies on p^* before node r. Suppose, contrary to our claim, that $\hat{r}$ lies on p^* after r (see Figure 1a). Then subpath

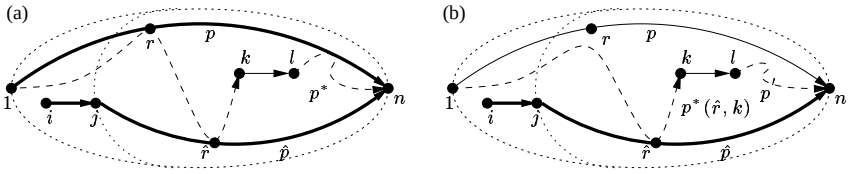

Fig. 1. (a) Configuration $\Omega^* - \hat{r}$ on p^* after r (b) Configuration $\Omega^{'}$ (activities with the maximal durations are in bold)

$p^*(\hat{r}, n)$, $p^*(\hat{r}, n) = p^*(\hat{r}, k) \cup p^*(k, n)$, is at least as long as subpath $\hat{p}(\hat{r}, n)$ in configuration Ω^*. Notice that $p^*(\hat{r}, k)$ is one of longest paths from $\hat{r}$ to k in Ω^*. We may now decrease some activity durations to their lower bounds in configuration Ω^* in the following way (see Figure 1b) $d_{uv}(\Omega^{'}) = \begin{cases} d^+_{uv} & \text{if } (u,v) \in \hat{p}, \\ d^+_{uv} & \text{if } (u,v) = (i,j), \quad (u,v) \in A. \\ d^-_{uv} & \text{otherwise,} \end{cases}$ Duration $d_{ij}(\Omega^{'}) = d^+_{ij}$, and, by assumption, (k, l) is necessarily critical in $G(D_{ij} = d^+_{ij})$. Consequently (k, l) is critical in this new configuration $\Omega^{'}$. Hence, there exists a critical path $p^{'} \in P$ traversing (k, l). Since node $\hat{r}$ lies on p^* after node r, $l_{p^*(\hat{r},k)}(\Omega^*) = l_{p^*(\hat{r},k)}(\Omega^{'})$. Therefore path $p^*(\hat{r}, k) \cup p^{'}(k, n)$ is at least as long as subpath $\hat{p}(\hat{r}, n)$ in configuration $\Omega^{'}$. Decreasing $d_{ij}(\Omega^{'})$ to its lower bound gives configuration $\Omega^+_{\hat{p}}$. Observe that the lengths of paths $p^*(\hat{r}, k) \cup p^{'}(k, n)$ and $\hat{p}(\hat{r}, n)$ remain unchanged. Hence, there exists the path in $G(j, n)$ composed of two subpaths $\hat{p}(j, \hat{r})$ and $p^*(\hat{r}, k) \cup p^{'}(k, n)$ that is at least as long as $\hat{p}$, which is impossible because we have assumed that no critical path goes through (k, l) in $G(j, n)$ in configuration $\Omega^+_{\hat{p}}$. We can now return to the main proof.

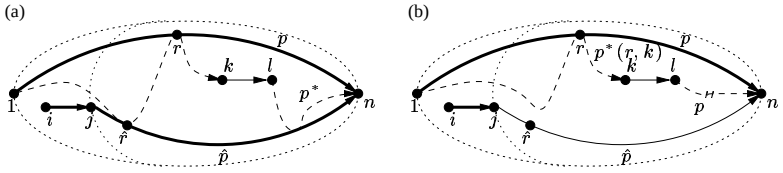

Fig. 2. (a) Configuration $\Omega^* - \hat{r}$ on p^* before r (b) Configuration $\Omega^{''}$

Consider configuration Ω^*. The claim shows that if node $\hat{r}$ exists, then $\hat{r} = r$ or node $\hat{r}$ lies on p^* before node r (see Figure 2a). In the case when $\hat{r}$ does not exist, the proof proceeds in the same manner. From the above and the criticality of p^* in Ω^*, it follows that subpath $p^*(r, n)$, $p^*(r, n) = p^*(r, k) \cup p^*(k, n)$, is at least as long as subpath $p(r, n)$ in this configuration. Notice that $p^*(r, k)$ is one of longest paths from r to k in Ω^*. Decreasing some of durations in Ω^* to their lower bounds, we obtain configuration $\Omega^{''}$

in the following form (see Figure 2b) $d_{uv}(\Omega'') = \begin{cases} d^+_{uv} & \text{if } (u,v) \in p, \\ d^+_{uv} & \text{if } (u,v) = (i,j), \quad (u,v) \in A. \\ d^-_{uv} & \text{otherwise,} \end{cases}$

Duration $d_{ij}(\Omega'') = d^+_{ij}$, by assumption, (k,l) is necessarily critical in $G(D_{ij} = d^+_{ij})$, which implies the criticality of (k,l) in this new configuration Ω''. Hence, there exists a critical path $p'' \in P$ using (k,l). By the claim, $l_{p^*(r,k)}(\Omega^*) = l_{p^*(r,k)}(\Omega'')$. It follows that path $p^*(r,k) \cup p''(k,n)$ is at least as long as subpath $p(r,n)$ in configuration Ω''. If $(i,j) \notin p$, we may decrease duration $d_{ij}(\Omega'')$ to its lower bound. Again by the claim, the lengths of subpaths $p^*(r,k) \cup p''(k,n)$ and $p(r,n)$ remain unchanged in this new configuration and so $p^*(r,k) \cup p''(k,n)$ is still at least as long as $p(r,n)$. It is easily seen that this new configuration is equal to Ω^+_p. If $(i,j) \in p$, configurations Ω''_p and Ω^+_p are equal. Consequently, path $p(1,r) \cup p^*(r,k) \cup p''(k,n)$ is at least as long as p and moreover $p(1,r) \cup p^*(r,k) \cup p''(k,n)$ uses (k,l). This contradicts our assumption that no critical path in Ω^+_p includes (k,l). □

We are now in a position to give an algorithm (Algorithm 2) for asserting necessary criticality of a fixed activity in a general network. At each step of the algorithm, tasks between j and k have precise durations (so Algorithm 1 can be invoked), and Algorithm 2 assigns precise durations to tasks preceding j, while preserving the criticality of task (k,l). Since Algorithm 1 runs in $O(m)$, Algorithm 2 requires $O(mn)$ time.

Algorithm 2:

Input: A network $G =< V, A >$, activity (k,l), interval durations D_{uv}, $(u,v) \in A$.
Output: `true` if (k,l) is necessarily critical in G; and `false` otherwise.
for $j \leftarrow k$ **downto** 2 **such that** $j \in PRED(k) \cup \{k\}$ **do**
  NC $\leftarrow$ Algorithm 1 with $G(j,n)$ and durations D_{uv};
  if NC =`true` **then foreach** $i \in Pred(j)$ **do** $D_{ij} \leftarrow d^-_{ij}$;
  else foreach $i \in Pred(j)$ **do** $D_{ij} \leftarrow d^+_{ij}$;
end
NC $\leftarrow$ Algorithm 1 with $G(1,n)$,(k,l) and durations D_{uv};
if NC =`true` **then return** `true`;
else return `false`;

Remark: Lemma 1, Propositions 2 and 3 have counterparts for asserting possible criticality. So the reasoning which leads to the first algorithm can be applied for possible criticality [13]. This leads to an algorithm similar to Algorithm 1 swapping durations d^-_{uv} and d^+_{uv}, for asserting the possible criticality of tasks whose predecessors have deterministic durations. Unfortunately, Propositions 4 and 5 can not be adapted to the study of possible criticality, and asserting if a task is possibly critical in the general case is provably NP-Complete [7]. So, while the results of this section are instrumental for solving problem LUBF, the same approach cannot be applied to compute the GLB of the floats.

5 Computing intervals for the latest starting times

5.1 Computing the Greatest Lower Bound on the Latest Starting Times

There already exists an algorithm that computes the GLB of latest starting times [13] in polynomial time whose basis is recalled below. This section presents a new polynomial method (more efficient than the already known one as shown in Section 7) derived from a path enumeration algorithm [8].

The Incremental Approach Let us recall the following simple but important result that allows to reduce the set of configurations $\mathfrak{C}$ for the GLBLST problem.

Proposition 6 ([11]) *The greatest lower bound on the latest starting times lst^-_{kl} of activity (k,l) in G is attained on an extreme configuration in which the duration of (k,l) is at its upper bound and all activities that do not belong to set $SUCC(k,l)$ have durations at their lower bounds.*

The idea of the algorithm for computing lst^-_{kl} is based on Lemma 2. It consists in finding the minimal nonnegative real number f^*_{kl} that added to the upper bound of the duration interval of a specified (k,l) makes it possibly critical.

Lemma 2 ([13]) *Let f^*_{kl} be the minimal nonnegative real such that (k,l) is possibly critical with a duration $d^+_{kl} + f^*_{kl}$. Then $lst^-_{kl} = est^-_{kl} + f^*_{kl}$.*

The sketch of the algorithm is simple: we begin to set the durations of tasks preceding (k,l) to their minimal value then we run the algorithm asserting the possible criticality of (k,l). If (k,l) is possibly critical then its minimal latest starting date equals the minimal earliest starting date. Otherwise for each task $(i,j) \in PRED(k,l)$ for which (k,l) is possibly critical in $G(j,n)$ and not in $G(i,n)$ we compute the minimal duration to add to (k,l) to make this task possibly critical in $G(i,n)$ (this value is easy to compute). Then we add the smallest computed value to the duration of (k,l) and reiterate the test of possible criticality of (k,l) in G. Contrary to the next one, this algorithm computes the latest starting date of only one task.

The Path Enumeration Approach First let us state a result which describes the form of configurations where the GLB of the latest starting date of a task (k,l) in a network G is attained, given in [8]. Let $P_{kl}(u,v)$ be the set of all paths from node u to node v going through task (k,l).

Proposition 7 *Let $(k,l) \in A$ be a task of G. There exists a path $p_{kl} \in P_{kl}(k,n)$ such that the extreme configuration $\Omega^+_{p_{kl}}$ minimizes $lst_{kl}(.)$.*

Note that path p_{kl} is one of the longest paths from k to n including l in configuration $\Omega^+_{p_{kl}}$. We can recursively construct the path $p_{kl} \in P_{kl}(k,n)$ of

Proposition 7 corresponding to the optimal configuration. Suppose that for each node $u \in Succ(l)$, we know a path $p_{lu} \in P_{lu}(l,n)$ for which the configuration $\Omega^+_{p_{lu}}$ minimizes the latest starting date of (l,u). Then we can construct an optimal path p_{kl} from paths p_{lu} for $u \in Succ(l)$.

Proposition 8 *Let (k,l) be a task of G, and $\forall u \in Succ(l)$, let $p_{lu} \in P_{lu}(l,n)$ be a path such that $lst^-_{lu} = lst_{lu}(\Omega_{p_{lu}})$. Then $lst^-_{kl} = \min_{u \in succ(l)} lst_{kl}(\Omega_{\{(k,l)\} \cup p_{lu}})$.*

From Proposition 8, we can deduce a polynomial algorithm to compute the GLB of the latest starting dates of all tasks: we recursively find a path p_{kl} for which the configuration $\Omega^+_{p_{kl}}$ minimizes $lst_{kl}(.)$ from the paths p_{lu} for $u \in Succ(l)$, starting from the nodes in $Pred(n)$. Algorithm 3 runs in $O(m(n+m))$ for the computation of all the GLBs of the latest starting dates. Note that it is similar to the backwards recursion technique used in the classical CPM method.

Algorithm 3:

Input: A network G, interval durations D_{uv}, $(u,v) \in A$;
Output: The GLB of latest starting dates of all the tasks in network;
foreach $(k,l) \in A$ **do** $lst^-_{kl} \leftarrow +\infty$;
$V \leftarrow V \cup \{n+1\}$; $A \leftarrow A \cup \{(n, n+1)\}$;
$D_{nn+1} \leftarrow 0$; $p_{nn+1} \leftarrow (n, n+1)$;
foreach (k,l) **such that** $k \leftarrow n-1$ **downto** 0 **do**
  foreach $u \in Succ(l)$ **do**
    $p' \leftarrow (k,l) \cup p_{lu}$;
    Compute $lst_{kl}(\Omega_{p'})$ by the classical CPM;
    if $lst_{kl}(\Omega_{p'}) < lst^-_{kl}$ **then** $lst^-_{kl} \leftarrow lst_{kl}(\Omega_{p'})$; $p_{kl} \leftarrow p'$;
  end
end

5.2 Computing the Least Upper Bound on the Latest Starting Times

Only a counterpart to the incremental method is known for computing the GLB on the latest starting times of an activity. There exists an exponential path enumeration algorithm [8], but it has not (yet) been adapted to compute the LUB of the latest starting dates in polynomial time (contrary to the GLB of the latest starting dates). Again a result of Dubois et al. [11], allows to reduce the set of configurations $\mathfrak{C}$.

Proposition 9 ([11]) *The least upper bound on the latest starting times lst^-_{kl} of activity (k,l) in G is attained on an extreme configuration in which the duration of (k,l) is at its lower bound and all activities that do not belong to set $SUCC(k,l)$ have durations at their upper bounds.*

The main idea of the algorithm for determining lst_{kl}^{+} of a given activity $(k,l) \in A$ is based on Lemma 3. It consists in determining the minimal nonnegative real number f_{kl}^{*} that added to the lower bound of the duration interval of a specified activity (k,l) makes it necessarily critical.

Lemma 3 ([13]) *Let f_{kl}^{*} be the minimal nonnegative real number such that (k,l) is necessarily critical with a duration $d_{kl}^{-} + f_{kl}^{*}$. Then $lst_{kl}^{+} = est_{kl}^{+} + f_{kl}^{*}$.*

6 Computational methods for floats

6.1 Computing the Least Upper Bound on Floats

To compute the LUB of the floats of an activity (k,l), we first set the durations of the tasks neither preceding nor succeeding (k,l) according to the following Lemma 4. The maximal float of (k,l) after this partial instantiation is the same as in the original network G.

Lemma 4 ([11]) *The least upper bound on float f_{kl}^{+} of activity (k,l) in G is attained on an extreme configuration in which the duration of (k,l) is at its lower bound and all activities from set $A \setminus SUCC(k,l) \setminus PRED(k,l) \setminus \{(k,l)\}$ have durations at their upper bounds.*

Algorithm 2 is a polynomial algorithm which can already assert if the task (k,l) is necessarily critical. To compute the LUB of the floats, we are going to increase step by step the duration of (k,l) from $D_{kl} = d_{kl}^{-}$ until (k,l) becomes necessarily critical. Lemmas 5 and 7 give the hint to find the increment of D_{kl} at each step of the algorithm. According to Proposition 10, this incremental technique eventually yields f_{kl}^{+}.

Lemma 5 *Let activities $(i,j) \in PRED(k,l)$ have precise durations in G. Then (k,l) is necessarily critical in G if and only if there exists a path $p \in P(1,k)$ such that for every node $j \in p$, (k,l) is necessarily critical in $G(j,n)$.*

Proof. ($\Longrightarrow$) Let us denote by p a longest path from 1 to k. Note that activities $(i,j) \in PRED(k,l)$ have precise durations. From the necessary criticality of (k,l) in G, it follows that p is part of a longest path from 1 to n and this path denoted by p uses (k,l) for each configuration. Thus for every node $j \in p$, the subpath $p(j,n)$ is critical path in $G(j,n)$. Since this is true for each configuration, (k,l) is necessarily critical in $G(j,n)$.
($\Longleftarrow$) Just take $j = 1$. $G(1,n) = G$ and so (k,l) is necessarily critical in G. □

If we assume that activities $(i,j) \in PRED(k,l)$ have precise durations, then there is a connection between the least upper bound on latest starting times lst_{kl}^{+} and the least upper bound on floats f_{kl}^{+} of activity (k,l).

Lemma 6 ([13]) *Let activities $(i,j) \in PRED(k,l)$ have precise durations in G. Then $f_{kl}^{+} = lst_{kl}^{+} - est_{kl}^{+}$.*

Accordingly, under the assumption that activities $(i,j) \in PRED(k,l)$ have precise durations, the least upper bound on floats f_{kl}^{+} can be computed by means of algorithms for determining lst_{kl}^{+} presented in Section 5. At each iteration of the while loop of Algorithm 4, (k,l) become necessarily critical in at least one new subnetwork $G(j,n)$. Thus the loop is executed at most n times, and so Algorithm 4 takes $O(n^3m)$ time.

Lemma 7 *Let* $\Delta = \min\{f_{kl}^{+}(j,n) \mid (k,l)$ *is not necessarily critical in* $G(j,n)\}$ *where* $f_{kl}^{+}(j,n)$ *is the least upper bound on floats in* $G(j,n)$, $j \in PRED(k)$. *Then for all* $\epsilon < \Delta$, *activity* (k,l) *is not necessarily critical in* $G(j,n)$, $j \in PRED(k)$, *with duration* $D_{kl} = d_{kl}^{-} + \epsilon$. *Moreover there exists* $j^* \in PRED(k)$ *such that* (k,l) *is not necessarily critical in* $G(j^*,n)$ *and* (k,l) *is necessarily critical in* $G(j^*,n)$ *with duration* $D_{kl} = d_{kl}^{-} + \Delta$.

Proof. By Observation 1, one can assume that activity (k,l) has the duration of the form $D_{kl} = [d_{kl}^{-}, d_{kl}^{-}]$.

Consider a node j such that (k,l) is not necessarily critical in $G(j,n)$. Let Ω be an extreme configuration where the float of (k,l) attains its maximal value $f_{kl}^{+}(j,n) > 0$ in $G(j,n)$, and $\epsilon < \Delta \leq f_{kl}^{+}(j,n)$ $(d_{kl}(\Omega) = d_{kl}^{-})$. p' denotes the longest path in $G(j,n)$ in Ω, while p'' stands for the longest path in $G(j,n)$ in Ω using (k,l). Therefore, $f_{kl}^{+}(j,n) = l_{p'}(\Omega) - l_{p''}(\Omega) > \epsilon$.

Now let us define the configuration Ω' such that $d_{uv}(\Omega') = \begin{cases} d_{kl}^{-} + \epsilon & \text{if } (u,v) = (k,l) \\ d_{uv}(\Omega) & \text{otherwise,} \end{cases}$.

p' remains a longest path in Ω' with same length, and p'' is still a longest path traversing (k,l) with lenght $l_{p''}(\Omega') = l_{p''}(\Omega) + \epsilon$. Since the float of (k,l) in Ω' is $l_{p'}(\Omega') - l_{p''}(\Omega') > 0$, (k,l) is not necessarily critical in $G(j,n)$, $j \in PRED(k)$, with duration $D_{kl} = d_{kl}^{-} + \epsilon$.

Consider a node j^* such that (k,l) is not necessarily critical in $G(j^*,n)$, and set $\Delta = f_{kl}^{+}(j^*,n)$. Then for all configurations, the difference between the longest path in $G(j^*,n)$ and the longest path using (k,l) is less or equal than Δ. If we increase the duration of (k,l) to $d_{kl}^{-} + \Delta$, the longest path, in this new configuration, will traverse (k,l) and thus (k,l) will be necessarily critical in $G(j^*,n)$ with duration $D_{kl} = d_{kl}^{-} + \Delta$. □

At the end of the incremental process, the least upper bound of the float of (k,l) is attained:

Proposition 10 *Let* f_{kl}^{*} *be the minimal nonnegative real number such that* (k,l) *is necessarily critical in* $G(D_{kl} = d_{kl}^{-} + f_{kl}^{*})$. *Then* $f_{kl}^{+} = f_{kl}^{*}$.

Proof. Consider any configuration Ω. Let p' be a longest path in Ω and p'' be a longest path including (k,l) in Ω. Note that $f_{kl}(\Omega) = l_{p'}(\Omega) - l_{p''}(\Omega)$. Let us now modify configuration Ω and denote it by Ω^x. Configuration Ω^x is defined as follows $d_{uv}(\Omega^x) = \begin{cases} d_{uv}(\Omega) + x & \text{if } (u,v) = (k,l) \\ d_{uv}(\Omega) & \text{otherwise,} \end{cases}$ $(u,v) \in A$,where x is a nonnegative real number. It is clear that $l_{p''}(\Omega^x) = l_{p''}(\Omega) + x$ and p'' remains a longest path including (k,l) in new configuration Ω^x. Consider the following two cases.

Case: $x < f_{kl}(\Omega)$. Then $l_{p''}(\Omega^x) \leq l_{p'}(\Omega)$, and so p' is still a critical path in Ω^x and has the same length as in Ω. This gives $f_{kl}(\Omega^x) = f_{kl}(\Omega) - x$.

Case: $x \geq f_{kl}(\Omega)$. Then p'' becomes a critical path and so $f_{kl}(\Omega^x) = 0$. Thus, for all

Ω and x equation $f_{kl}(\Omega^x) = \max(f_{kl}(\Omega) - x, 0)$ holds. In particular, for $x = f^*_{kl}$ and configuration $\Omega = \Omega^*$ such that Ω^* maximizes the float of task (k,l) in G. From the definition of f^*_{kl} we get $f_{kl}(\Omega^{f^*_{kl}}) = 0$, hence $f_{kl}(\Omega^*) - f^*_{kl} \leq 0$, and finally $f^+_{kl} \leq f^*_{kl}$.

Suppose that $f^+_{kl} < f^*_{kl}$. Set $y = (f^*_{kl} + f^-_{kl})/2$. Then, for every Ω, $f^+_{kl}(\Omega^y) = \max(f_{kl}(\Omega) - y, 0) = 0$. Note that y is nonnegative real number, smaller than f^*_{kl}, such that (k,l) is necessarily critical in $G(D_{kl} = d^-_{kl} + y)$, which contradicts with the definition of f^*_{kl}. Hence, we conclude that $f^+_{kl} = f^*_{kl}$. □

Algorithm 4:

Input: A network G, activity (k,l), interval durations D_{uv}, $(u,v) \in A$.
Output: The least upper bound on floats f^+_{kl}.
NC $\leftarrow$ Algorithm 2; /*Set task durations preceding (k,l) according Prop. 4,5 */
$f^+_{kl} \leftarrow 0$; $d_{kl} \leftarrow d^-_{kl}$;
while NC =`false` **do**
  $\Delta \leftarrow \min\{f^+_{kl}(j,n) \mid j \in PRED(k),\ f^+_{kl}(j,n) \neq 0\}$; /*$f^+_{kl}(j,n)$ in $G(j,n)$ is computed according to Lemma 6 */
  $f^+_{kl} \leftarrow f^+_{kl} + \Delta$; $d_{kl} \leftarrow d_{kl} + \Delta$;
  NC $\leftarrow$ Algorithm 2; /*Update precise durations of tasks preceding (k,l) */
end
return f^+_{kl};

6.2 Computing the greatest lower bound on floats

Computing GLB on floats is NP-Hard in the general case. The reader should refer to [8, 13] for some special tractable cases. However, an efficient algorithm has been proposed in [8]. The idea is to compute a PERT/CPM on each configuration Ω_p such that p is a path of G from 1 to n. The number of tested configuration is of course potentially exponential, but in practice the algorithm runs very fast on realistic problems.

7 Complexity and Experimental Results

First, we summarize in Table 1 the complexity levels of the different problems of the PERT/CPM on intervals. Moreover, in Table 1, we give the complexity of the best known algorithm which computes the quantity of interest. In particular, the path algorithm which computes the GLB and LUB of floats and latest starting dates has a complexity in $O((n+m)|P|)$, where $|P|$ is the number of path of the graph G. This complexity depends of the topology of the network.

We now present some computational results in order to evaluate the performance of all these algorithms on scheduling problems library of 600 networks of 120 tasks (PSPLIB: http://halfrunt.bwl.uni-kiel.de/bwlinstitute/Prod/psplib/). Those instances of problems have been generated by the ProGen program, for activity network generation [14]. In Table 2, we give the minimal, maximal and

Table 1. The complexity of the interval problems and the best known algorithms running times

Earliest starting date (all tasks)	GLB	P	$O(n+m)$
	LUB	P	$O(n+m)$
Latest starting date (one task)	GLB	P	$O(mn)$
	LUB	P	$O(mn)$
Latest starting date (all tasks)	LUB	P	$O(m(m+n))$
Float (all tasks)	GLB	NP-Hard	$O((n+m)\|P\|)$
Float (one task)	LUB	P	$O(n^3m)$

average execution times (in second) of five algorithms on those 600 problems. GLBLST and LUBLST are the polynomial algorithms which compute the latest starting dates given in [13]. As it can be seen from the experimental results, the

Table 2. The minimal, maximal and average execution times (in second) of five algorithms on 600 problems

	Min	Max	Average
Path algorithm [8] LUBLST, GLBLST, LUBF, GLBF	0.02	0.45	0.12
Polynomial path algorithm (Algorithm 3) LUBLST	0.01	0.03	0.017
GLBLST [13]	0.01	0.05	0.023
LUBLST [13]	0.02	0.12	0.056
LUBF (Algorithm 4)	0.57	8.2	3.12

path algorithm (potentially exponential) is very efficient in practice to compute at the same time the latest starting dates and the floats. On the contrary, the polynomial algorithm which computes the LUB of the floats is not efficient in practice for realistic scheduling problems. However, its performance should be better for larger instances.

8 Conclusion

This paper has proposed a complete solution to the criticality analysis of a network of activities when durations of tasks are ill-known. It is shown that moving from precise to imprecise durations radically changes the complexity of the problem, ruining the traditional critical path method. Part of the problem, pertaining to the GLB of the float, becomes NP-hard, the other questions remaining polynomial, although not straightforwardly so. These complexity results shed light on reasons why the more familiar stochastic counterparts to this basic scheduling problem are so difficult to handle, part of the difficulty being already present

in the interval analysis. The proposed algorithms can be of obvious relevance to practical predictive project scheduling problems where durations of tasks are not known in advance. Clearly, due to the basic nature of the addressed problem, several lines of research can be envisaged for future research. For instance one may assume that part of the tasks durations are controllable and additional constraints relating durations and starting or ending times may be available. Then one obtains a makespan minimization problem in the more general setting of STPU's. Another interesting question is to relate the above results to robust scheduling when several scenarios are available. In the latter case scenarios embody dependencies between task durations while our approach makes no dependence assumptions. Robust scheduling becomes more and more difficult as the number of scenarios increases. In this case our approach may provide a good approximation if duration intervals are derived from a large number of scenarios. Alternatively robust and interval scheduling can be hybridized considering a small set of imprecise (interval-valued) scenarios derived from the large scenario set by means of a clustering method.

References

1. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial intelligence **49** (1991) 61–95
2. Morris, P., Muscettola, N., Vidal, T.: Dynamic control of plans with temporal uncertainty. In: IJCAI. (2001) 494–502
3. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: from consistency to controllabilities. JETAI **11** (1999) 23–45
4. Morris, P.H., Muscettola, N.: Managing temporal uncertainty through waypoint controllability. In: IJCAI. (1999) 1253–1258
5. Khatib, L., Morris, P., Morris, R., Rossi, F.: Temporal constraint reasoning with preferences. In: IJCAI. (2001) 322–327
6. Kelley, J., Walker, M.: Critical path planning and scheduling. In: Proc. of the Eastern Joint Comp. Conf. (1959) 160–172
7. Chanas, S., Zieliński, P.: The computational complexity of the criticality problems in a network with interval activity times. EJOR **136** (2002) 541–550
8. Dubois, D., Fargier, H., Fortin, J.: Computational methods for determining the latest starting times and floats of tasks in interval-valued activity networks. J. of Intelligent Manufacturing (2005) To appear.
9. Buckley, J.: Fuzzy PERT. In: Applications of fuzzy set methodologies in industrial engineering. Elsevier (1989) 103–114
10. Dubois, D., Fargier, H., Fortemps, P.: Fuzzy scheduling: modeling flexible constraints vs. coping with incomplete knowledge. EJOR **147** (2003) 231–252
11. Dubois, D., Fargier, H., Galvagnon, V.: On latest starting times and floats in activity networks with ill-known durations. EJOR **147** (2003) 266–280
12. Chanas, S., Dubois, D., Zieliński, P.: On the sure criticality of tasks in activity networks with imprecise durations. IEEE Transactions on Systems, Man, and Cybernetics **34** (2002) 393–407
13. Zieliński, P.: On computing the latest starting times and floats of activities in a network with imprecise durations. Fuzzy Sets and Systems **150** (2005) 53–76
14. Kolisch, R., Sprecher, A.: Psplib - a project scheduling library. EJOR **96** (1996) 205–216