Temporal Analysis of Projects Under Interval Uncertainty

Christian Artigues, Cyril Briand, Thierry Garaix

Chapter 1
Temporal Analysis of Projects Under Interval Uncertainty

Christian Artigues, Cyril Briand and Thierry Garaix

Abstract Given an activity-on-node network where every activity has an uncertain duration represented by an interval, this chapter takes an interest in computing the minimum and maximum earliest start times, latest start times and floats of all activities over all duration scenarios. The basic results from the literature are recalled and efficient solving algorithms are detailed. A particular focus is put on the computation of minimum float, which remains a $\mathscr{NP}$-hard optimization problem. For this last case, a recent and efficient branch and bound algorithm is described that outperforms previously proposed methods.

Key words: Project scheduling, Temporal analysis, Interval uncertainty, Algorithms, Complexity, Branch-and-bound.

1.1 Introduction

In standard deterministic project scheduling, temporal analysis aims at determining the temporal degree of freedom of activities under simple finish-start precedence constraints. More precisely, it aims at computing for every activity i its earliest start and completion times ES_i and EC_i, its latest start and completion times LS_i and LC_i and its total float TF_i. It is well known that these values can be computed via longest path computation in the project network where each arc (i, j) is evaluated by the duration of i. More precisely, if d_{ij} denotes the length of the longest path from i to j in this graph, ES_i is the longest path from dummy node 0 to node i: $ES_i = d_{0i}$. LS_i is the length of the longest path from dummy node 0 do dummy node $n+1$ (schedule

Christian Artigues and Cyril Briand
CNRS LAAS, Université de Toulouse, 31400, Toulouse, France, e-mail: {artigues,briand}@laas.fr

Thierry Garaix
CIS ROGI-LIMOS UMR CNRS 6158, École Nationale Supérieure des Mines de Saint-Étienne, 42000, Saint-Étienne, France, e-mail: garaix@emse.fr

length or makespan) minus the length of the longest path from node i to node $n+1$: $LS_i = d_{0n+1} - d_{in+1}$. The total float can be defined as the difference between LS_i and ES_i or, equivalently, $TF_i = d_{0n+1} - d_{0i} - d_{in+1} = LS_i - ES_i$. Standard longest path computations in acyclic graph allow to compute all these values in $\mathcal{O}(|E|)$ time, E being the set of precedence constraints.

When problem parameters are ill-known, a common way of modeling uncertainty is to define each such parameter as an interval, as discussed in Chap. **??**. In this chapter, we consider that the duration of each activity $i \in V$ is defined as an interval $[p_i^{\min}, p_i^{\max}]$. Under such an assumption, temporal analysis now focuses on the computation of the minimum and maximum values of the earliest start times, latest start times and total floats.

Let us define the scenario set Σ as the set of possible duration vectors:

$$\Sigma = \{\sigma \in \mathbb{R}^n | p_i^{\min} \leq p_i \leq p_i^{\max}, \forall i \in V\}$$

The temporal analysis of a project network under interval uncertainly consists in computing the following values for each activity $i \in V$:

- Best-case (minimum) earliest start time $ES_i^{\min} = \min_{\sigma\in\Sigma} ES_i(\sigma)$.
- Worst-case (maximum) earliest start time $ES_i^{\max} = \max_{\sigma\in\Sigma} ES_i(\sigma)$.
- Best-case (maximum) latest start time $LS_i^{\max} = \max_{\sigma\in\Sigma} LS_i(\sigma)$.
- Worst-case (minimum) latest start time $LS_i^{\min} = \min_{\sigma\in\Sigma} LS_i(\sigma)$.
- Best-case (maximum) float $TF_i^{\max} = \max_{\sigma\in\Sigma} TF_i(\sigma)$.
- Worst-case (minimum) float $TF_i^{\min} = \min_{\sigma\in\Sigma} TF_i(\sigma)$.

These values are of interest for providing valuable information to the decision-maker about the level of criticality of the activities despite the uncertain nature of the processing times. Any activity i such that $TF_i^{\max} = 0$ is necessarily critical and should consequently be carefully monitored independently of the scenario. Any activity i such that $TF_i^{\min} = 0$ is possibly critical and a special attention should be paid to it, especially for risk-adverse decision policies. On the contrary, if $TF_i^{\min} > 0$, the information that the activity has flexibility for all scenarios is obtained.

Table 1.1 provides the minimum and maximum earliest start times, latest start times and floats for the project network displayed in Fig. 1.1.

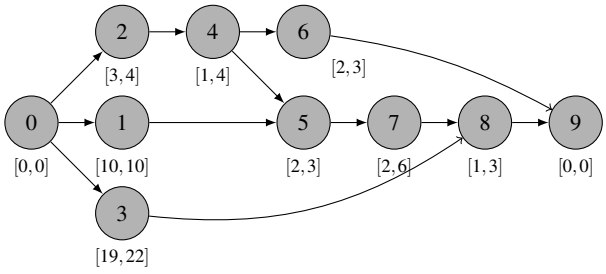

Fig. 1.1 Project network with interval uncertainty

Table 1.1 Minimum and maximum earliest start times, latest start times and floats

i	1	2	3	4	5	6	7	8	9
$[ES_i^{\min}, ES_i^{\max}]$	[0,0]	[0,0]	[0,0]	[3,4]	[10,10]	[4,8]	[12,13]	[19,22]	[20,25]
$[LS_i^{\min}, LS_i^{\max}]$	[0,8]	[2,14]	[0,0]	[6,17]	[10,18]	[17,23]	[13,20]	[19,22]	[20,25]
$[TF_i^{\min}, TF_i^{\max}]$	[0,8]	[2,14]	[0,0]	[2,14]	[0,8]	[9,19]	[0,8]	[0,0]	[0,0]

We see in Table 1.1 that some non-dummy activities are necessarily critical (activities 3 and 8) while others are possibly critical (activities 1, 5 and 7). We also remark that intuition does not necessarily work to obtain the minimum total float. In fact we have some cases where $TF_i^{\max} \neq LS_i^{\max} - ES_i^{\min}$ and others where $TF_i^{\min} \neq LS_i^{\min} - ES_i^{\max}$.

Several authors have studied these temporal analysis problems. The chapter is based on the results obtained by Chanas et al. (2001, 2002), Chanas and Zieliński (2002), Chanas and Zielinski (2003), Dubois et al. (2003, 2005), Fargier et al. (2000), Fortin et al. (2010), Kasperski and Zieliński (2010), Zieliński (2003, 2005, 2006), Garaix et al. (2013). Among these works we refer in particular to the survey by Fortin et al. (2010) in which it is also remarked that the interval problems are particular cases of fuzzy project scheduling problems (see Chaps. **??**). Sect. 1.2 summarizes the main structural properties complexity results and proposed solution algorithms detailed by Fortin et al. (2010). Sect. 1.3 focuses on algorithms for the minimum and maximum latest start time problems. Sect. 1.4 presents the algorithms for solving the minimum and maximum float problems. Finally, Sect. 1.5 provides concluding remarks and direction for future work in connection with related problems, such as controlability of simple temporal networks with uncertainty.

Let us precise that in this chapter, in contrast with some of the above-cited papers, the formalism on Activity-On-Node graph is used instead of the one of Activity-On-Arc (AOA).

1.2 Basic Properties, General Algorithms and Complexity Results (Fortin et al. 2010)

1.2.1 Extreme Scenarios

Let us define the notion of extreme scenario induced by an activity subset. Let $\sigma^{\max}(Q)$, with $Q \subseteq V$, the extreme scenario such that each activity $i \in Q$ is set to $p_i^{\max}$ while each activity of $i \in V \setminus Q$ is set to $p_i^{\min}$. Remark that the minimum and maximum earliest start times of each activity $i \in V$ are attained on extreme scenarios induced by the empty set and V, respectively:

$$ES_i^{\min} = ES_i(\sigma^{\max}(\emptyset))$$

$$ES_i^{\max} = ES_i(\sigma^{\max}(V))$$

For determining the latest start times and floats, the solution is not so trivial. However Dubois et al. (2003) showed that the searched minimum and maximum values are attained for extreme scenarios.

Theorem 1.1 (Dubois et al. 2003). *For every activity $i \in V$:*

$$LS_i^{\max} = \max_{Q \subseteq V} LS_i(\sigma^{\max}(Q))$$

$$LS_i^{\min} = \min_{Q \subseteq V} LS_i(\sigma^{\max}(Q))$$

$$TF_i^{\max} = \max_{Q \subseteq V} TF_i(\sigma^{\max}(Q))$$

$$TF_i^{\min} = \min_{Q \subseteq V} TF_i(\sigma^{\max}(Q))$$

Proof. To understand these properties, let us illustrate them on the minimum float case. Suppose that the minimum float of an activity i is reached on a scenario σ that is not extreme. Let P denote the longest path passing through i on scenario σ. Suppose in addition that among all scenarios yielding the minimum float for i, σ is such that the position on P of the first activity j not set to its largest or smallest duration is minimum. First, let us change scenario σ in scenario σ' by modifying the duration of all activities – except those located on P– by switching them to their minimum duration. Obviously the longest path in the network cannot be increased by this operation and P remains one of the longest path passing through i. It follows that the float of i cannot be increased by this modification of σ. We further modify σ' by increasing j to its maximum duration so as to obtain scenario σ''. This obviously increases the longest path passing through i by $p'_j - p_j^{\min}$. This also potentially increases the longest path in the network, but at most by $p'_j - p_j^{\min}$. Hence the float of i cannot be decreased by this last change, which contradicts the minimality of the position of j in P. □

A purely enumerative algorithm would then enumerate all 2^n extreme scenarios and, for each of them, compute the required longest paths. In the 8 non-dummy activities example of Fig. 1.1, these would yield to 256×2 longest path computations.

1.2.2 Path-induced Extreme Scenarios

Dubois et al. (2005) further restricted the search space by establishing the following properties on extreme scenarios induced by paths. They first show that the minimum and maximum floats, as well as the maximum latest start time of any activity are always attained for an extreme scenario induced by the activities located on a (0–$n+1$) path. Let $\mathscr{P}_{ij}$ denote the set of paths from activity i to activity j. By a slight abuse of notation, we identify the set of activities located on a path P by the path itself.

Theorem 1.2 (Dubois et al. 2005). *For every activity $i \in V$:*

$$LS_i^{\max} = \max_{P \in \mathscr{P}_{0n+1}} LS_i(\sigma^{\max}(P))$$

$$TF_i^{\max} = \max_{P \in \mathscr{P}_{0n+1}} TF_i(\sigma^{\max}(P))$$

$$TF_i^{\min} = \min_{P \in \mathscr{P}_{0n+1}} TF_i(\sigma^{\max}(P))$$

Proof. Observing the proof of Theorem 1.1, we can only change the definition of σ as an optimal extreme scenario not induced by a path and of j as the activity located in P with minimal position such that $p_j \neq p_j^{\max}$. Then the modifications change P into an optimal path where j is set to $p_j^{\max}$. □

Then to compute the maximum latest start times, the minimum and the maximum float, one could enumerate all paths in $\mathscr{P}_{0n+1}$, and for each of them, obtain the induced extreme scenario and compute the longest path lengths.

In the example from Fig. 1.1, the number of longest path computations boils down to $|\mathscr{P}_{09}| \times 2 = 8$. Although the number of paths from 0 to $n+1$ is always smaller than 2^n and can be small for sparse graphs, it can also be unfortunately exponential in n, as illustrated in the network given in Fig. 1.2 where the number of paths from 0 to $n+1$ is equal to $2^{\frac{n}{2}}$.

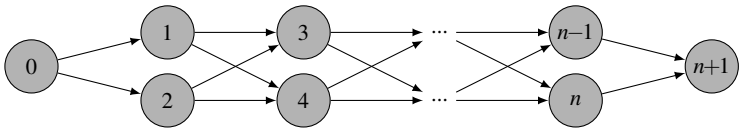

Fig. 1.2 Project network with an exponential number of paths

The reader may have noticed that the minimum latest start time was excluded from Theorem 1.2. For this value, Dubois et al. (2005) show that one can restrict the search to scenarios induced by a path from i to $n+1$.

Theorem 1.3 (Dubois et al. 2005). *For every activity $i \in V$:*

$$LS_i^{\min} = \min_{P \in \mathscr{P}_{in+1}} LS_i(\sigma^{\max}(P))$$

Proof. The proof proceeds the same way as for Theorems 1.1 and 1.2. Suppose σ is an optimal extreme scenario not induced by a $(i$–$n+1)$ path and let P denote the longest path from i to $n+1$ for scenario σ. Setting all activities in $V \setminus P$ to their minimum duration cannot increase the latest start time. Setting then all activities duration of P to their maximum duration can only decrease or leave unchanged the latest start time. □

There can still be an exponential number of paths in $\mathscr{P}_{in+1}$. However Fortin et al. (2010) provided a dynamic programming recursion allowing to compute the optimal

path of an activity i in a polynomial time (see Sect. 1.3.2), given the optimal paths of its direct successors.

Can we obtain the same positive complexity results for the maximum latest start time, the minimum and maximum floats?

1.2.3 Complexity Results

Actually, asserting the necessary criticality of an activity requires $\mathscr{O}(|E||V|)$ time using the algorithm proposed in Fortin et al. (2010), while the asserting the possibly criticality is strongly $\mathscr{NP}$-Complete (Chanas and Zieliński 2002). However, both problems were shown to be polynomial by Zieliński (2005) in $\mathscr{O}(|E|+|V|)$ time, when durations of the predecessors of the activities are precisely known (Zieliński 2005).

Even if the minimum float problem is polynomial for series-parallel graphs (Fargier et al. 2000, Zieliński 2006), these positive complexity results cannot be extended to this problem for general graphs. The minimum float problem was indeed proven to be strongly $\mathscr{NP}$-hard and even has no polynomial approximation (Chanas et al. 2002, Chanas and Zieliński 2002, Zieliński 2005).

1.2.4 Link with Min Max Regret Longest Path Problems

Let us point out that the minimum float problem can be linked to the min max regret longest path problem in acyclic graphs. Let P be a path between 0 and $n+1$. Given a scenario σ, the regret of P is the difference between the length of the longest path in the network for scenario σ and the length of P for scenario σ. The min max regret longest path problems amounts to find a path P^* from 0 to $n+1$ that minimizes this maximum regret.

If there exists a scenario for which the min max regret is zero, we have found a scenario for which P is critical, which amounts to find a set of possibly critical activities, with a zero minimum float. Conversely if we have a path of necessarily critical activities (of zero maximum floats), we have found a critical path for all scenarios and consequently a path of zero min max regret.

In the remaining of this chapter, we describe the polynomial algorithms for computing the minimum and maximum latest start times. We also describe a branch and bound algorithm that performs remarkably well, compared to the path algorithm, for the minimum float problem.

1.3 Maximum and Minimum Latest Start Time

In Zieliński (2005), two similar polynomial algorithms are given to compute the minimum and maximum latest start times of an activity. They are respectively linked to two underlying problems; asserting on possibly and necessary criticality of an activity when durations of its predecessors are fixed. For the case of minimum latest start time, another recursive algorithm is proposed in Fortin et al. (2010). In this section, we propose a slightly different description of these algorithms, for the sake of clarity and concision.

1.3.1 Maximum Latest Start Time

Theorem 1.4 allows to restrict the number of scenarios to consider during the search of the maximum latest start time of an activity i. We refer to $\overline{Succ}(i)$ ($\overline{Pred}(i)$) as the set of all immediate and transitive successors (predecessors, respectively) of each activity i.

Theorem 1.4. *The set of scenarios* $\Sigma^{i,\max} = \{\sigma \in \Sigma | p_j(\sigma) = p_j^{\max}, \forall j \notin \overline{Succ}(i) \cup \{i\}\}$ *is dominant when the maximum latest start time of activity* $i \in V$ *is seek.*

Proof. The proof is straightforward as the latest start time of an activity LS_i, $i \in V$, is defined by the gap between d_{0n+1} and d_{in+1}. Transformations of a scenario by switches from $p_j^{\min}$ to $p_j^{\max}$ on activities $j \notin \overline{Succ}(i) \cup \{i\}$ can only increase this gap and, finally, reach a scenario of $\Sigma^{i,\max}$. □

In order to describe the algorithm of Zieliński (2005), it is necessary to link the maximum latest start time of an activity to its the necessary criticality. Let $\Sigma^{i,\max}(p_i = v)$ the subset of scenarios of $\Sigma^{i,\max}$ where in addition $p_i = v$, v being any value in $[p_i^{\min}, p_i^{\max}]$.

Theorem 1.5. $LS_i^{\max} = ES_i^{\max} + \Delta$, *where* $\Delta = \min\{\delta \in \mathbb{R} | i$ *is necessary critical in* $\{\sigma \in \Sigma^{i,\max}(p_i = p_i^{\min} + \delta)\}\}$, $\forall i \in V$.

Proof. It is enough to show that $ES_i^{\max} + \Delta = ES_i(\sigma) + \Delta = LS_i(\sigma)$ for all scenarios of $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$. For any $\sigma \in \Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$, $LS_i(\sigma)$ can not be strictly greater than $ES_i(\sigma) + \Delta$, by definition of necessary criticality of i. A strictly lower value of $LS_i(\sigma)$ contradicts the minimality of Δ. □

The main idea of Algorithm 1 is to iteratively increment the duration of i until reaching the minimum value $(p_i^{\min} + \Delta)$, such that i becomes necessarily critical, the considered set of scenarios being restricted to $\Sigma^{i,\max}$. Note that for large enough values of Δ, $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$ does not include feasible scenarios since p_i can become greater than $p_i^{\max}$. The tricky point is to define the increment step at each iteration. That is done by considering activities $j \in \overline{Succ}(i)$ for which i is not *j-critical*, this concept being defined in Theorem 1.6.

Theorem 1.6. *Activity i is not necessary critical in $\sigma \in \Sigma^{i,\max}(p_i(\sigma) = \bar{p})$, if and only if there exists at least one activity $j \in \overline{Succ}(i)$ on each longest path from i to $n+1$ such that $\delta_j(\sigma) = d_{0j}(\sigma) - (d_{0i}(\sigma) + d_{ij}(\sigma)) > 0$, i.e., the longest path from 0 to j traversing i is not a longest path from 0 to j in σ. Activity i is said j-critical when $\delta_j(\sigma) = 0$.*

Proof. The proof is directly derived from the definition of the criticality of an activity. □

According to Theorems 1.5 and 1.6, for any duration $\bar{p} < p_i^{\min} + \Delta$, there exists at least one scenario $\sigma \in \Sigma^{i,\max}(p_i(\sigma) = \bar{p})$ and one activity $j \in \overline{Succ}(i)$ such that i is not j-critical ($\delta_j(\sigma) > 0$). Then an increase of the duration of i by the minimum $\delta = \min_{j \in \overline{Succ}(i)} \delta_j(\sigma)$ will decrease by at least one (j itself) the number of activities for which i is not j-critical in each scenario. The algorithm ends when i becomes j-critical for all activities of $j \in \overline{Succ}(i)$, and so necessary critical as expected in Theorem 1.5.

Algorithm 1 Computing $LS_i^{\max}$

1: $\Delta := 0$;
2: $\delta := min_{j \in \overline{Succ}(i)}\{\delta_j(\sigma)\}$ on $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$;
3: **if** $\delta > 0$ **then**
4: $\quad \Delta := \Delta + \delta$ and **goto** Step 2;
5: **end if**
6: $LS_i^{\max} := ES_i^{\max} + \Delta$;

At the initialization phase, the maximum earliest start time of i and the minimum earliest start times of activities $\overline{Succ}(i)$ in scenarios $\Sigma^{i,\max}(p_i = p_i^{\min})$ can be computed, as a preprocessing phase, through a PERT algorithm under extreme scenario $\sigma^{\max}\left(V \setminus (\overline{Succ}(i) \cup \{i\})\right)$. The procedure called at Step 2 of Algorithm 1, which computes values $\delta_j(\sigma)$, is detailed in Algorithm 2. Since the topological order is followed, the value of $\delta_j(\sigma)$ only depends on (predecessor) activities with fixed durations; initialized at Step 1 and updated at Steps 4-8 of Algorithm 2. By construction (Steps 4-8), the gap between $d_{0j}(\sigma)$ and $d_{0i}(\sigma) + d_{ij}(\sigma)$, is decreased as much as possible for next activities $k \in \overline{Succ}(j)$. Thus, each computed value of $\delta_j(\sigma)$ is maximal on $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$. We highlight that $\delta_j(\sigma)$ can be viewed as the total float of activity i, under scenario $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$, in the subgraph involved by $\overline{Pred}(j)$ with $j \in \overline{Succ}(i)$. The evaluation of $\delta_j(\sigma)$ can be done in constant time as lengths of partial longest paths $d_{0j}(\sigma)$ and $d_{ij}(\sigma)$ can be dynamically updated. Therefore, Algorithm 2 runs in $\mathcal{O}(|E|)$ time and Algorithm 1 in $\mathcal{O}(|V||E|)$ time.

Note that Algorithm 2 allows to assert the necessary criticality of an activity when durations of its predecessors are precisely known (Zieliński 2005) .

Algorithm 2 Computing $\delta_j(\sigma), \forall j \in \overline{Succ}(i)$ on $\Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$

1: Set partial scenario $\sigma \in \Sigma^{i,\max}(p_i = p_i^{\min} + \Delta)$;
2: $j :=$ next activity of $\overline{Succ}(i)$ according to the topological order;
3: $\delta_j(\sigma) := d_{0j}(\sigma) - d_{0i}(\sigma) - d_{ij}(\sigma)$;
4: **if** $\delta_j(\sigma) > 0$ **then**
5: $\quad p_j(\sigma) := p_j^{\max}$;
6: **else**
7: $\quad p_j(\sigma) := p_j^{\min}$;
8: **end if**
9: **if** $j = n+1$ **then**
10: $\quad$ **stop**;
11: **else**
12: $\quad$ **goto** Step 2;
13: **end if**

1.3.2 Minimum Latest Start Time

Theorem 1.3 gives graph-topological properties that reduce the search space to longest paths from i to $n+1$. The recursion procedure of Theorem 1.7 allows to compute the optimal path from i to $n+1$.

Theorem 1.7 (Fortin et al. 2010). *For each activity $i \in V$:*

$$LS_i^{\min} = \min_{j \in Succ(i)} LS_i(\sigma^{\max}(\{i\} \cup Q_j)) \text{ with } Q_j \in \mathscr{P}_{in+1}, LS_j^{\min} = LS_j(\sigma^{\max}(Q_j))$$

From this recursion, as a unique path can be computed for each successor, a polynomial algorithm can be obtained. Fortin et al. (2010) proposed an $\mathscr{O}((|E|+|V|)^2)$ algorithm. This shows that the minimum latest start time computation is polynomial despite the possibly exponential number of paths in $\mathscr{P}_{in+1}$.

Another $\mathscr{O}((|E|+|V|)^2)$ time algorithm has been previously proposed in Zieliński (2005). Algorithm 3 is similar to the algorithm that computes the maximum latest start time of an activity but it is based on the relation between the minimum latest start time and the possible criticality of an activity.

The dominant set of scenarios is denoted $\Sigma^{i,\min}$ and is defined by setting durations of activities of $V \setminus \{\overline{Succ}(i) \cup \{i\}\}$ to their respective lower bounds. This result is justified in the argument of Theorem 1.3.

Here Δ represents the minimum value to add to the duration of i to make it possibly critical, i.e., j-critical in at least one scenario. This procedure is given in Algorithms 3 and 4. The main difference with Algorithms 1 and 2, which gives $LS_i^{\max}$, is the updating step of partial scenario σ. The gap between $d_{0k}(\sigma)$ and $d_{0i}(\sigma) + d_{ik}(\sigma)$ for next activities $k \in \overline{Succ}(i)$, can only be decreased by Steps 4-8 of Algorithm 4.

Again, one can derive an algorithm, presented in Zieliński (2005), which allows to assert the possibly criticality of an activity when its predecessors have fixed durations.

Algorithm 3 Computing $LS_i^{\min}$

1: $\Delta := 0$;
2: $\delta := \min_{j \in \overline{Succ}(i)} \{\delta_j(\sigma)\}$ on $\Sigma^{i,\min}(p_i = p_i^{\max} + \Delta)$;
3: **if** $\delta > 0$ **then**
4: $\quad \Delta := \Delta + \delta$ and **goto** Step 2;
5: **end if**
6: $LS_i^{\min} := ES_i^{\min} + \Delta$;

Algorithm 4 Computing $\delta_j(\sigma), \forall j \in \overline{Succ}(i)$ on $\Sigma^{i,\min}(p_i = p_i^{\max} + \Delta)$

1: Set partial scenario $\sigma \in \Sigma^{i,\min}(p_i = p_i^{\max} + \Delta)$;
2: $j :=$ next activity of $\overline{Succ}(i)$ according to the topological order;
3: $\delta_j(\sigma) := d_{0j}(\sigma) - d_{0i}(\sigma) - d_{ij}(\sigma)$;
4: **if** $\delta_j(\sigma) > 0$ **then**
5: $\quad p_j(\sigma) := p_j^{\min}$;
6: **else**
7: $\quad p_j(\sigma) := p_j^{\max}$;
8: **end if**
9: **if** $j = n+1$ **then**
10: $\quad$ **stop**;
11: **else**
12: $\quad$ **goto** Step 2;
13: **end if**

1.4 Minimum and Maximum Floats

1.4.1 Maximum Floats

In Zieliński (2005)), the author proposed a polynomial time algorithm able to determine whether an activity i is critical. It is based on two properties. The first one states that if i is necessarily critical in a subgraph made of the activities of $\overline{Succ}(j)$, for some $j \in \overline{Pred}(i)$, then i is necessarily critical in the general graph if and only if there exists a scenario σ with $p_j(\sigma) = p_j^{\min}$ such that $TF_i(\sigma) = 0$. The second one claims that if i is *not* necessarily critical in the subgraph made of the activities of $\overline{Succ}(j)$, for some $j \in \overline{Pred}(i)$, then i is necessarily critical in the general graph if and only if there exists a scenario σ with $p_j(\sigma) = p_j^{\max}$ such that $TF_i(\sigma) = 0$. These properties allow to derive Algorithm 5 that asserts the necessary criticality of activity i. The algorithm is called with the initial scenario $\sigma^{\min}$. At the first iteration (i.e., $j = i$), it first calls Algorithm 2 (see Step 2) to determine whether i is necessarily critical in the graph made of the activities of $\overline{Succ}(i)$. Thus it is possible to fix the durations of the activities immediately preceding i, with respect to both above properties, without modifying the necessary criticality of i (at Steps 3-7). We can reiterate this processus considering the activities of $\overline{Pred}(i)$ according to the reverse topological order (see Step 1). Once the durations of the activities belonging to $\overline{Pred}(i)$ are totally set, i is necessarily critical if and only if, at the last iteration, $TF_i(\sigma) = 0$. Algorithm 5 asserts the necessarily criticality of i in $\mathcal{O}((|E| + |V|)^2)$

time. Nevertheless, let us highlight that in the case where i is not necessarily critical the, the value $TF_i(\sigma) > 0$ eventually found gives a lower bound of $TF_i^{\max}$.

Algorithm 5 Asserting the necessary criticality of activity i

1: $j :=$ next activity of $\{\overline{Pred}(i) \cup \{i\}\}$ according to the *reverse* topological order;
2: Compute $\delta = TF_i(\sigma)$ in the subgraph $\overline{Succ}(j)$ using Algorithm 2;
3: **if** $\delta > 0$ **then**
4: $\quad p_j(\sigma) := p_j^{\max}$;
5: **else**
6: $\quad p_j(\sigma) := p_j^{\min}$;
7: **end if**
8: **if** $j \neq 0$ **then**
9: $\quad$ **goto** Step 1
10: **end if**;

The algorithm that computes the maximum float of activities is based on the previous algorithm and uses the following property.

Theorem 1.8 (Zieliński 2005). *If Δ is the smallest positive real value such that i becomes necessarily critical in scenario σ such that $p_i(\sigma) = p_i^{\min} + \Delta$, then $TF_i^{\max} = \Delta$.*

Proof. The argument is based on the fact that, whatever the consider scenario $\sigma \in \Sigma$, any longest path traversing i remains a longest path if p_i is increased by a small value. If Δ is the minimum value to add to p_i such that i become necessarily critical (i.e., $TF_i(p_i^{\min} + \Delta) = 0$) then, because $TF_i(p_i^{\min} + \Delta) \leq TF_i^{\max}$, it easy to deduce the claimed property. □

The problem is now to find Δ. The idea is to use a similar approach that the one used for computing $LS_i^{\max}$ using the property (proved in Zieliński 2003) that $TF_i^{\max} = LS_i^{\max} - ES_i^{\max}$ in the case where the durations of the activities belonging to $\overline{Pred}(i)$ are known (that property being falsed in the general case). So, it becomes possible to determine the smallest δ such that i becomes necessarily critical in a subgraph made by the activities of $\overline{Succ}(j)$, with $j \in \overline{Pred}(i)$.

The above idea is implemented in Algorithm 6, which works as follows. A first assignment of the durations of the activities $\overline{Pred}(i)$ is made by Algorithm 5 in Step 3. If $TF_i(\sigma) = 0$ then i is necessarily critical. Otherwise, for every $j \in \overline{Pred}(i)$ and for the scenario σ determined by Algorithm 5 (i.e., the durations of $\overline{Pred}(i)$ are known), the lower bound of the maximum float $LS_i^{\max}(\sigma) - d_{0i}(\sigma)$ is computed in the graph $\overline{Succ}(j)$ using Algorithm 1 (see Step 7). The value δ of the smallest positive lower bound is then memorized (see Steps 8-10). Finally, Δ and p_i are incremented by δ in Steps 14-15 and the value of $TF_i(\sigma)$ is recomputed. The algorithm ends when i becomes critical in σ and $TF_i^{\max} = \Delta$. This algorithm works in $\mathcal{O}((|E| + |V|)^4)$.

Algorithm 6 Computing $TF_i^{\max} = \Delta$

$p_j(\sigma) := p_j^{\min}, \forall j \in V \setminus \{\overline{Pred}(i) \cup \overline{Succ}(i)\}$;
$\Delta := 0$;
Compute $TF_i(\sigma)$ and update σ with respect to decisions made in Algorithm 5;
while $TF_i(\sigma) > 0$ **do**
  $\delta := +\infty$;
  $j :=$ previous activity of $\overline{Pred}(i)$ according to the reverse topological order;
  Compute $LS_i^{\max}(\sigma)$ in the subgraph $\overline{Succ}(j)$ using Algorithm 1;
  if $LS_i^{\max}(\sigma) - d_{0,i}(\sigma) > 0$ **then**
    $\delta_j(\sigma) := \min(\delta, LS_i^{\max}(\sigma) - d_{0,i}(\sigma))$;
  end if
  if $j \neq 0$ **then**
    goto Step 6
  end if;
  $\Delta := \Delta + \delta$;
  $p_i(\sigma) := p_i^{\min} + \Delta$;
  Compute $TF_i(\sigma)$ and update σ with respect to decisions made in Algorithm 5;
end while

1.4.2 Minimum Floats

This subsection focuses on the computation of the minimum float TF_i^{min} of every activity i, which is an $\mathscr{NP}$-hard problem (Chanas et al. 2002, Chanas and Zieliński 2002, Zieliński 2005). From Theorem 1.2, we know that any optimum is obtained for a particular scenario $\sigma^{\max}(P)$ with $P \in \mathscr{P}_{0n+1}$. Note that P is also the longest path from 0 to $n+1$ traversing i. Using this property, Dubois et al. (2005) and Fortin et al. (2010) proposed a first exact algorithm based on path enumeration, the float of activities being computed for every generated path using standard PERT method. Typically, this algorithm is able to compute in a few seconds the minimum float $TF_i^{\min}$ of an activity on medium-density graph with 100 activities. However, its performance gets worse on high density graph since the number of paths to explore grows exponentially.

A faster branch-and-bound procedure was recently proposed in Garaix et al. (2013). It takes benefits from other problem properties. This procedure is able to compute minimum floats within few milliseconds for 100-activity graph, this CPU-time remaining rather insensitive to graph density variation. We review in this section the basic ingredients of the Garaix et al. (2013) procedure: a dominance property and a bounding rule.

Let us first take an interest in the structure of the longest path obtained for a path-induced extreme scenario $\sigma^{\max}(P)$.

The dominance Theorem 1.9 below states that the longest path $P' \in \mathscr{P}_{0n+1}$ in any optimal scenario $\sigma^{\max}(P)$ differs from P only by a subpath $P'_{a\to b}$, a being a predecessor of i in P, and b a successor of i in P. Moreover, in the particular case where $TF_i^{\min} = 0$ then, since P and P' are identical, $a = b = i$. This property is illustrated in Fig. 1.3 where the arcs in bold define the longest path P traversing i inducing the extreme scenario. The thin arc $P'_{a\to b}$ represents the deviation from P

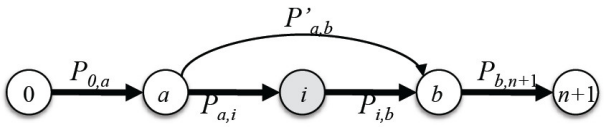

Fig. 1.3 Dominant path structures for float computation

of the longest path P' that results from the concatenation of three partial paths (i.e., $P' = (P_{0\to a}, P'_{a\to b}, P_{b\to n+1})$). Furthermore, the length of $P'_{a\to b}$ equals the length of the longest path from a to b in the scenario $\sigma^{\min}(V)$, plus $p_a^{\max} - p_a^{\min}$ (the duration of activity a being set to $p_a^{\max}$ as it belongs to P).

Theorem 1.9 (Garaix et al. 2013).

For any activity $i \in V$, there exists an optimal path P with scenario $\sigma^{\max}(P)$ and a pair of activities $a,b \in P$ such that the longest path P' in this scenario is $P' = (P_{0\to a}, P'_{a\to b}, P_{b\to n+1})$ verifying

$$TF_i^{min} = d_{ab}(\sigma^{\min}(V)) + p_a^{\max} - p_a^{\min} - \sum_{k\in P_{a\to b}\setminus\{b\}} p_k^{\max}$$

Proof. The proof of this theorem goes by showing that if $P_{0\to a}$ is not a longest path from 0 to a then there exists an alternative path $P'_{0\to a}$ such that $TF_i(\sigma^{\max}(P)) \geq TF_i(\sigma^{\max}(P'_{0\to a}, P_{a\to b}, P_{b\to n+1}))$. Similarly, if $P_{b\to n+1}$ is not a longest path from b to $n+1$ then there exists another scenario path-induced scenario leading to a lower float for i. □

A major interest of Theorem 1.9 is to formalize the intuitive fact that any path-induced scenario $\sigma^{\max}(P)$ such that the subpath from 0 to a is not also a longest path from 0 to a can be discarded (since it is dominated with respect to the minimization of TF_i). Symmetrically, any path $P \in \mathscr{P}$ whose subpath from b to $n+1$ is not also a longest path from b to $n+1$ under scenario $\sigma^{\max}(P)$, is also dominated. In the sequel, any path which cannot be discarded in this way will be said *valid* with respect to the minimization of TF_i.

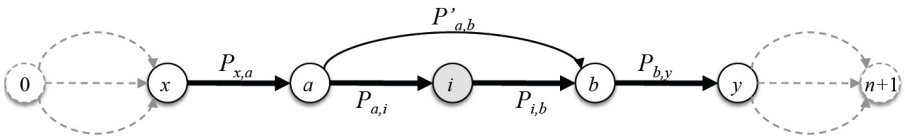

Fig. 1.4 Partial path structure for float computation

Let us take an interest now in the computation of lower bounds for $TF_i^{\min}$. For that purpose, let us consider a valid *partial* path $P_{x\to y}$ with $i \in P_{x\to y}$ (see Fig. 1.4). We highlight that $P_{x\to y} = (P_{x\to a}, P_{a\to i}, P_{i\to b}, P_{b\to y})$ is said valid in the sense that $P' = (P_{x\to a}, P'_{a\to b}, P_{b\to y})$ is the longest path in the scenario $\sigma^{\max}(P_{x\to y})$ between x and y.

From the optimality principle of Bellman it is easy to show that the following theorem holds:

Theorem 1.10 (Garaix et al. 2013).
Considering a valid partial *path $P_{x\to y}$ with $i \in P_{x\to y}$, if there exists a valid path $P_{0\to n+1}$ extending $P_{x\to y}$ towards activities* 0 *and* $n+1$*, it satisfies:*

$$TF_i(\sigma^{\max}(P)) \geq LB_i(P_{x\to y}),$$

with

$$LB_i(P_{x\to y}) = d_{ab}(\sigma^{\min}(V)) + p_a^{\max} - p_a^{\min} - \sum_{k\in P_{a\to b}\setminus\{b\}} p_k^{\max}.$$

Proof. Considering a valid path extension $P_{0\to n+1}$ of a partial path $P_{x\to y}$, we know from Theorem 1.9 that there exists a^* and $b^* \in V$ such that $TF_i(\sigma^{\max}(P)) = d_{a^*b^*}(\sigma^{\min}(V)) + p_{a^*}^{\max} - p_{a^*}^{\min} - \sum_{k\in P_{a^*\to b^*}\setminus\{b^*\}} p_k^{\max}$. From Bellman's optimality principle, any path from 0 to $n+1$ diverging from P from an activity $a \neq a^*$ and converging back to P on an activity $b \neq b^*$ has a length lower or equal to the longest one passing through a^* and b^*. From this latter property, it is easy to deduce the inequality claimed in Theorem 1.10. □

In other words, whatever the considered possible valid extension P of $P_{x\to y}$, Theorem 1.10 ensures that $TF_i(\sigma^{\max}(P))$ will never be lower than $LB_i(P_{x\to y})$. This value actually corresponds to the float of i under the scenario $\sigma^{\max}(P_{x\to y})$ when only the activities belonging to $(P_{x\to y} \cup P'_{a\to b})$ are considered.

Theorems 1.9 and 1.10 allow to design an efficient branch-and-bound procedure for the computation of the minimum floats. In this procedure, the nodes of the search tree correspond to valid partial paths related to a given activity i, which are stored inside a stack $\mathscr{Q}$ for depth-first search. Given a valid partial path $P_{x\to y}$ with $i \in P_{x\to y}$, the branching scheme consists in alternatively extending the path to the left or to the right, by considering either all the immediate predecessors of x, or all the immediate successors of y, discarding the non-valid path extensions (with respect to the dominance rule). Thus, considering a given path-extension direction δ, a node has always as many children as immediate valid path-extensions. Each node of the search tree also memorizes the direction δ ($\delta \in \{\text{left, right}\}$) to consider for the next path extension. A leaf of the search tree corresponds to a valid path P from 0 to $n+1$ and is evaluated by $TF_i(\sigma^{\max}(P))$. Classically, a partial path $P_{x\to y}$ is deleted only if its current evaluation, $LB_i(P_{x\to y})$, is greater than the best float TF_i already found.

Let us comment on Algorithm 7. We remark first that the longest path values $d_{ij}(\sigma^{\min}(V))$ are precomputed using the variant of Bellman-Ford's algorithm for DAGs. Computing the minimum float of all activities requires running the branch-and-bound procedure n times (see Step 1). At the beginning of each run (Steps 2-3), TF_i is set to ∞, stack $\mathscr{Q}$ only contains a single partial path $P_{x\to y} = i$, and the path-extension direction is set to left by default. The branch-and-bound procedure is implemented in Steps 4-34. While stack $\mathscr{Q}$ is not empty, a partial path $P_{x\to y}$ is taken from the stack, with its path-extension direction δ (Step 5). Preliminarily, the

Algorithm 7 Branch-and-bound

1: **for all** $i \in V \setminus \{0, n+1\}$ **do**
2: $TF_i := \infty$;
3: push($\mathscr{P}$, $(i$,left$)$);
4: **while** $\mathscr{P} \neq \emptyset$ **do**
5: $(p_{x\to y}, \delta)$:=pop($\mathscr{P}$);
6: $(a,b) := \underset{\{u\in p_{x\to i}, v\in p_{i\to y}\}}{\text{argmax}} d_{u,v}(\sigma^{\min}(V)) + p_u^{\max} - p_u^{\min} - \sum_{k\in P_{u\to v}\setminus\{v\}} p_k^{\max}$;
7: **if** $d_{a,b}(\sigma^{\min}(V)) + p_a^{\max} - p_a^{\min} - \sum_{k\in P_{a\to b}\setminus\{b\}} p_k^{\max} < TF_i$ **then**
8: **if** $x = 0$ AND $y = n+1$ **then**
9: $TF_i := d_{a,b}(\sigma^{\min}(V)) + p_a^{\max} - p_a^{\min} - \sum_{k\in P_{a\to b}\setminus\{b\}} p_k^{\max}$;
10: **else if** δ =left **then**
11: **for all** $x' \in \Gamma^{-1}(x)$ **do**
12: **if** $p_{x'}^{\max} + \sum_{k\in P_{x\to u}\setminus\{u\}} p_k^{\max} \geq d_{x',u}(\sigma^{\min}(V)) + p_{x'}^{\max} - p_{x'}^{\min}, \forall u \in P_{x\to i}$ **then**
13: **if** $y \neq n+1$ **then**
14: δ :=right;
15: **else**
16: δ :=left;
17: **end if**
18: push($\mathscr{P}$, $((x', P_{x\to y}), \delta)$);
19: **end if**
20: **end for**
21: **else if** δ =right **then**
22: **for all** $y' \in \Gamma(y)$ **do**
23: **if** $p_{y'}^{\max} + \sum_{k\in P_{v\to y}\setminus\{v\}} p_k^{\max} \geq d_{v,y'}(\sigma^{\min}(V)) + p_v^{\max} - p_v^{\min}, \forall v \in p_{i\to y}$ **then**
24: **if** $x \neq 0$ **then**
25: δ :=left;
26: **else**
27: δ :=right;
28: **end if**
29: push($\mathscr{P}$, $((P_{x\to y}, y'), \delta)$);
30: **end if**
31: **end for**
32: **end if**
33: **end if**
34: **end while**
35: **end for**

activities (a,b) for which the longest path from x to y differs from $p_{x\to y}$ are updated (see Step 6). Note that this can be done incrementally in linear time: if x (y) is the last activity toward which the path has been extended, only pairs (u,v) such that $u = x$ ($v = y$) and $v \in P_{i\to y}$ ($u \in P_{x\to i}$) are considered, respectively.

If $TF_i \leq LB_i(P_{x\to y})$, the path is deleted (see Step 7). Otherwise, if $P_{x\to y} \in \mathscr{P}_{0n+1}$, the new TF_i value is memorized (see Steps 8-9). If $P_{x\to y} \notin \mathscr{P}_{0n+1}$, it is extended with respect to direction δ. Below, only the case δ=left is commented on (see Steps 10-20), the case δ =right (see Steps 21-32) being symmetrical.

All the immediate predecessors x' of x are first considered for possible path extension $(x', P_{x\to y})$ (see Step 11). With respect to Theorem 1.9 , Step 12 verifies that it does not exist any activity $u \in P_{x\to i}$ such that the path $(x', P_{x\to u})$ has a smaller length than the one of the longest path between x' and u, otherwise the path extension is

not valid. We underline that the validity of a path extension can be checked in linear time since all longest path at minimum duration have been precomputed. Once a new left-path-extension is found, the next extension direction is set to right unless $y = n + 1$ (see Steps 13-18).

1.5 Conclusions

Computing the minimum and maximum values for the starting times and floats of project activities is a major concern of project managers. This assertion remains particularly valid when activity durations are modelled as intervals. Indeed, uncertain durations bring the concepts of possible and necessary activity criticality. This chapter showed how the necessary criticality can be checked in polynomial time, while the possible criticality remains $\mathscr{NP}$-hard to assert in general. An effective branch-and-bound procedure is proposed to cope with this last problem, which is able to compute the minimum float of the project activities.

It has already been pointed out by Fortin et al. (2010) the close connections of the criticality analysis presented in this chapter with fuzzy PERT scheduling problems on the one hand and min-max regret longest ath problems on the other hand. We also mention here that a closely related model, the simple temporal networks under uncertainty (STNU) have also been studied in the artificial intelligence community, see e.g. Morris et al (2001). Actually STNUs can be seen as a generalization of the activity network with uncertain interval duration to generalized precedence constraints, i.e., where arcs can have negative value representing a maximum time lag between two time points. The research that is mainly done in STNU is to assert dynamic controlability, which is roughly the ability of defining a schedule for any uncertain scenario. The criticality analysis presented in this chapter could be of interest to provide additional information on STNUs. A step towards such generalizations has recently been made by Yakhchali and Ghodsipour (2010).

References

Chanas S, Dubois D, Zieliński P (2001) Criticality analysis in activity networks under incomplete information. In: Proceedings of 2nd Int Conf Eur Soc of Fuzzy Logic and Applications, pp 233–236

Chanas S, Dubois D, Zieliński P (2002) On the sure criticality of tasks in activity networks with imprecise durations. IEEE T Syst Man Cy B 32(4):393–407

Chanas S, Zieliński P (2002) The computational complexity of the criticality problems in a network with interval activity times. Eur J Oper Res 136(3):541–550

Chanas S, Zieliński P (2003) On the hardness of evaluating criticality of activities in a planar network with duration intervals. Oper Res Lett 31(1):53–59

Dubois D, Fargier H, Galvagnon V (2003) On latest starting times and floats in activity networks with ill-known durations. Eur J Oper Res 147:266–280

Dubois D, Fargier H, Fortin J (2005) Computational methods for determining the latest starting times and floats of tasks in interval-valued activity networks. J Int Man 16(4–5):07–421

Fortin J, Zieliński P, Dubois D, Fargier H (2010) Criticality analysis of activity networks under interval uncertainty. J Sched 13(6):609–627

Fargier H, Galvagnon V, Dubois D (2000) Fuzzy PERT in series-parallel graphs. In 9th IEEE int conf on fuzzy syst, San Antonio, pp 717–722

Garaix T, Artigues C, Briand C (2013) Fast minimum float computation in activity networks under interval uncertainty. J Sched 16(1):93–103

Kasperski A, Zieliński P (2010) Minmax regret approach and optimality evaluation in combinatorial optimization problems with interval and fuzzy weights. Eur J Oper Res 200:680–687

Morris P, Muscettola N, Vidal T (2001) Dynamic control of plans with temporal uncertainty. In Nebel B. (ed) Proceedings of 17th Int Joint Conf Artificial Intelligence, Seattle, pp 494–499

Yakhchali S, Ghodsipour S (2010). Computing latest starting times of activities in interval-valued networks with minimal time lags. Eur J Oper Res, 200(3):874–880

Zieliński P (2003) Latest starting times and floats of activities in network with uncertain durations. 3rd Int Conf Eur Soc of Fuzzy Logic and Applications (EUSFLAT 2003), pp 586–591

Zieliński P (2005) On computing the latest starting times and floats of activities in a network with imprecise durations. Fuzzy Set Syst 150(1):53–76

Zieliński P. (2006) Efficient computation of project characteristics in a series-parallel activity network with interval durations. In Della Riccia G, Dubois D, Kruse R, Lenz HJ (ed), CISM courses and lectures: vol. 482. Decision theory and multi-agent planning. Springer, New York, pp 111–130