On computing the latest starting times and floats of activities in a network with imprecise durations

Pawel Zieliński

Słowa kluczowe:
planowanie, analiza sieciowa, liczba przedziałowa, liczba rozmyta, analiza krytyczności

Wrocław 2003
On computing the latest starting times and floats of activities in a network with imprecise durations

Paweł Zieliński
Institute of Mathematics,
Wrocław University of Technology,
Wybrzeże Wyspiańskiego 27,
50-370 Wrocław, Poland
pziel@im.pwr.wroc.pl

Abstract

The paper deals with the problems of computing the intervals of possible values of the latest starting times and floats of activities in networks with imprecise durations, represented by means of interval or fuzzy numbers. So far, these problems have been completely solved when the networks are series parallel. We propose new polynomial algorithms for determining the intervals of the latest starting times in general networks. We also present some complexity results for floats (the computation of floats is probably intractable) and describe some polynomially solvable cases. Then we extend the results to networks with fuzzy duration times.

Keywords: Project management and scheduling; Data intervals; Critical path analysis

1 Introduction

In this paper we wish to investigate the problems of computing the intervals (bounds) of possible values of the latest starting times and floats of activities in networks with uncertain durations modeled by fuzzy or interval numbers. These problems have attracted a considerable attention since the late 70’s, particularly because of their importance in project scheduling. So far attempts to solve the problems have been mainly based on the Critical Path Method [16] with formulas for the forward and the backward recursions, where the crisp arithmetic is replaced with the interval (fuzzy) arithmetic.

For such straightforward extension of CPM method, it turns out that the forward recursion correctly computes the sets of possible values of the earliest starting times of activities ([4], [8], [12]), but the backward recursion, with interval minimum and subtraction, fails to compute the sets of possible values of the latest starting times of activities ([19], [18], [21]). These times turn more and more imprecise while getting closer to the end of the calculation. Sometimes these times may be interval numbers with negative elements. So, the intervals of floats can no longer be recovered from

This work was supported by grant no. 7T11F02120 from the State Committee for Scientific Research (Komitet Badań Naukowych).
the intervals containing the earliest and latest starting times. This anomaly is caused
by an interactivity of interval variables. Several authors tried to cope with these prob-
lems (see [3] for a brief survey and a full bibliography). Kaufmann and Gupta [15],
Hapke et al. [14] and Rommelfanger [22] suggest substitutes to the interval (fuzzy)
subtraction, so as to improve the situation, but these techniques remain ad hoc. Naus-
tion [20] resorts to symbolic computations on the variable duration times. However,
this technique is unwieldy and highly combinatorial. The first attempt to obtain a
correct solution has been made in [2]. There has been provided the a possibilistic rep-
resentation of the problem of determining the fuzzy latest starting times of activities
and their floats, a difficulty connected to it has been pointed out, but without proposing
any solution methods. At present these problems have been completely solved when
the networks are series parallel (see [11], [9]). There have been provided \( O(n) \) algo-
rithms for both problems. In [9] some heuristic methods for computing the intervals of
possible values of the latest starting times and floats of activities in general networks
have been proposed. Here, we give \( O(mn) \) algorithms for computing the intervals of
possible values of the latest starting times of activities in general networks. We also
present complexity results for floats (the computation of floats is probably intractable)
and describe some polynomially solvable cases.

The paper is organized as follows. Section 2 formally defines the problems that
we consider. Section 2.1 gives an \( O(mn) \) time algorithm for computing the lower
bound on the latest starting times of a single activity in a general network. Section 2.2
gives an \( O(mn) \) time algorithm for computing the upper bound on the latest starting
times of a single activity in a general network. Section 2.3 shows that the problem
of determining the bounds on the float of an activity cannot be approximated within
a factor smaller than 1 and describes some polynomially solvable cases. Section 3
extends the results to the networks with fuzzy duration times. Section 4 concludes the
paper.

2 The latest starting times and floats of activities in a net-
work with duration intervals

A network \( G = \langle V, A \rangle \), being a project activity-on-arc model, is given. \( V \) is the set
of nodes (events), \( |V| = n \), and \( A \) is the set of arcs (activities), \( |A| = m \). The network
\( G \) is a directed, connected and acyclic graph. The set \( V = \{1, 2, \ldots, n\} \) is labeled
in such a way that \( i < j \) for each activity \( (i, j) \in A \). Weights of the arcs (activity
durations) \( (i, j) \in A \) are to be chosen from intervals \( T_{ij} = [t_{ij}, T_{ij}] \), \( T_{ij} \geq 0 \), two
nodes 1 and \( n \) are distinguished as the initial and final node, respectively.

We introduce some additional notations., which will be helpful in formulating and
proving results formulated in the paper.

- \( T \) denotes a configuration of activity durations \( t_{ij} \in T_{ij}, (i, j) \in A \), while
  \( t_{ij}(T) \) denotes the duration of activity \( (i, j) \) in configuration \( T \).

- \( \mathcal{T} \) is the set of possible configurations of the activity durations, i.e. \( \mathcal{T} \) is the
  Cartesian product of corresponding intervals \( T_{ij}, (i, j) \in A \).

- \( P \) is the set of all paths in \( G \) from node 1 to node \( n \).
2.1 Problem of evaluating the possible criticality of an activity – a polynomially solvable case

Let us recall the notions of the possible criticality of activities and paths in network $G$.

**Definition 1.** An activity $(k, l) \in A$ (resp. a path $p \in P$) is possibly critical in $G$ if and only if there exists a configuration of times $T \in \mathcal{T}$ such that $(k, l)$ (resp. $p$) is
critical in $G$ in the usual sense, after replacing the time intervals $T_{ij}$ by exact values $t_{ij}(T)$, $(i,j) \in A$.

The possible criticality have been thoroughly investigated in [5], [6]. The problem of the possible criticality for a path is polynomially solvable. Unfortunately, the same problem for an activity turns out to be strongly $NP$-complete for general networks and remains $NP$-complete even when a network is restricted to be planar (see [7]). In [11] a polynomial algorithm has been provided only in case of series-parallel networks.

The following proposition is obvious.

**Proposition 1.** An activity $(k,l) \in A$ is possibly critical in $G$ if and only if it belongs to some possibly critical path $p \in P$.

Lemma 1 gives necessary and sufficient conditions for establishing the possible criticality of a given path $p \in P$.

**Lemma 1 ([6]).** A path $p \in P$ is possibly critical in $G$ if and only if

$$t_{ij}(T) = \begin{cases} 0 & \text{for } (i,j) \in p, \\ t_{ij} & \text{for } (i,j) \notin p. \end{cases}$$

Let us focus on a network $G$, in which some duration intervals are precisely given. It is assumed that activities $(i,j) \in PREC(k,l)$ have duration times such that $t_{ij} = \bar{t}_{ij}$. We show that in this case the problem of evaluating the possible criticality of activity $(k,l) \in A$ is polynomially solvable. For such a network we define the set of possible configurations of activity durations and denote it by $\Xi^s(x)$, $x \geq 0$. $\Xi^s(x)$ is the Cartesian product of time intervals $T_{ij}(x)$, $(i,j) \in A$, given as follows:

$$\Xi^s_{ij}(x) = \begin{cases} [\bar{t}_{ij}, \bar{t}_{ij}] & \text{for } (i,j) \in SUCC(k,l), \\ [x, x] & \text{for } (i,j) = (k,l), \\ [\bar{t}_{ij}, \bar{t}_{ij}] & \text{otherwise.} \end{cases}$$

Similarly, we define the set of possible configurations of activity durations $\Xi^p(x)$ if activities $(i,j) \in SUCC(k,l)$ have precise durations. $\Xi^p(x)$ is the Cartesian product of $T_{ij}(x)$, $(i,j) \in A$, given as follows:

$$\Xi^p_{ij}(x) = \begin{cases} [\bar{t}_{ij}, \bar{t}_{ij}] & \text{for } (i,j) \in PREC(k,l), \\ [x, x] & \text{for } (i,j) = (k,l), \\ [\bar{t}_{ij}, \bar{t}_{ij}] & \text{otherwise.} \end{cases}$$

**Proposition 2.** Let activities $(i,j) \in PREC(k,l)$ (resp. $(i,j) \in SUCC(k,l)$) have precise duration times. Activity $(k,l)$ is possibly critical in $G$ if and only if there exists a configuration $T \in \Xi^s(\bar{t}_{kl})$ (resp. $T \in \Xi^p(\bar{t}_{kl})$) such that $(k,l)$ is critical in the usual sense for $T$ in $G$.

**Proof.** It follows directly from Definition 1, Propositions 6 and 7 and the fact that activities $(i,j) \in PREC(k,l)$ (resp. $(i,j) \in SUCC(k,l)$) have duration times such that $\bar{t}_{ij} = \bar{t}_{ij}$.

\[\square\]
Hence the problem of evaluating the possible criticality of \((k, l)\) in \(G\), in which the durations of activities \((i, j) \in P_{REC}(k, l)\) (resp. \((i, j) \in S_{UCC}(k, l)\)) are precisely given, boils down to the one in \(G\), in which the durations of activities \((i, j) \in A \setminus S_{UCC}(k, l)\) (resp. \((i, j) \in A \setminus P_{REC}(k, l)\)) are fixed (see (1), (2) and Proposition 2).

Now we give a polynomial algorithm (Algorithm 1) for asserting whether an activity \((k, l) \in A\) is possibly critical in \(G\), under the assumption that activities \((i, j) \in S_{UCC}(k, l)\) have precise duration times. The logic of Algorithm 1 is to construct a possibly critical path \(p \in P\) that uses \((k, l)\) or equivalently to find a configuration \(T' \in \bigoplus^{2}(\overline{T}_{kl})\) in which \((k, l)\) is critical in the usual sense according to Proposition 1.

To construct a possibly critical path (or find a configuration) a node labeling is performed with convenient setting of the activity durations. A node \(i \in V\) is labeled \(true\) if the longest path from node 1 to \(i\) uses activity \((k, l)\) in a systematically constructed configuration \(T\), and \(false\) otherwise. The duration time of activity \((i, j) \in A\) is set to its upper bound if node \(i\) is labeled \(true\), and to its lower bound otherwise, except for \((k, l)\). In this case the duration time of \((k, l)\) is set to its upper bound. It is worth pointing out that the algorithm is crucial and it will be substantially used in a algorithm for solving PLBLST. We will illustrate it with an example in Section 2.1.2.

The following lemma justifies Algorithm 1.

**Lemma 2.** Let activities \((i, j) \in P_{REC}(k, l)\) have precise duration times. Activity \((k, l)\) is possibly critical in network \(G\) if and only if Algorithm 1 returns \(PossCritical = true\) for \((k, l)\).

**Proof.** \((\Rightarrow)\) Assume that the activity \((k, l)\) is possibly critical in \(G\) in which activities \((i, j) \in P_{REC}(k, l)\) have precise duration times. Then there exists a possibly critical path \(p^* \in P(k, l)\) (see Proposition 1). Hence the path \(p^*\) is critical in the usual sense in \(G\) in the configuration \(T'' \in \bigoplus^{2}(\overline{T}_{kl})\) determined as in Lemma 1 (see Proposition 2). Consequently the length of \(p^*\) fulfills the following inequality

\[
l_{p^*}(T'') \geq l_p(T'), \quad \text{for all } p \notin P(k, l).
\]  

(3)

We claim that Algorithm 1 returns \(PossCritical = true\) for \((k, l)\). To prove this, assume to the contrary that Algorithm 1 returns \(PossCritical = false\). Note that it also returns \(label(i), i = 1, \ldots, n\). Let \(T'''' \in \bigoplus^{2}(\overline{T}_{kl})\) be a configuration of activity durations after the termination of Algorithm 1. \(PossCritical = false\), and so \(label(l) = false\) or \(label(n) = false\).

Consider the case when \(label(l) = false\). Then, \((k, l)\) does not lie on any critical path in configuration \(T''''\), \(t^*_i(T'') \neq t^*_k(T'') + t_{kl}(T'')\) (see Algorithm 1, rows: 9, 10). From this and the fact that \(t_{ij}(T') = t_{ij}(T'')\) for \((i, j) \notin S_{UCC}(k, l), T', T'''' \in \bigoplus^{2}(\overline{T}_{kl})\), it follows that there exists a path in \(T'\), which is longer then \(p^*\), contrary to (3). This contradicts that \(p^*\) is critical in \(T'\).

Consider the case when \(label(n) = false\). Then, \(label(l) = true\). Let us choose the subpath of \(p^*\) from \(k\) to \(r\), we denote it by \(p^*_k\), whose nodes \(i \in p^*_k\) are labeled \(true\) except for start node \(k\) and end node \(r\) (\(label(k) = false\) and \(label(r) = false\)). Such subpath exists, since \(p^* \in P\) and \(label(n) = false\). Hence duration times \(t_{ij}(T'')\) of activities \((i, j) \in p^*_k\) are equal to \(\overline{T}_{ij}\) (see Algorithm 1, rows: 13, 22). The result is \(t_{ij}(T'') = t_{ij}(T')\) for \((i, j) \notin S_{UCC}(k, l)\). Since \(label(r) = false\), there exists a path \(p_{1r}\) from 1 to \(r\) that does not use \((k, l)\),
Algorithm 1 Asserting whether an activity is possibly critical

Require: A network $G = \langle V, A \rangle$, a specified activity $(k, l) \in A$, time intervals $T_{ij} = [L_{ij}, T_{ij}]$, $(i, j) \in A$.

Ensure: A configuration $T \in \sum(\mathcal{T}_{kl})$, $PossCritical = true$ if $(k, l)$ is possibly critical, $false$ otherwise.

\begin{align*}
\triangleright \text{PHASE 1:} \\
1: & t_0^2 = 0; \text{label}(1) \gets false; \\
2: & \text{for } j \leftarrow 2 \text{ to } l - 1 \text{ do} \\
3: & \hspace{1em} \text{for all } i \in \text{Prec}(j) \text{ do} \\
4: & \hspace{2em} t_{ij} \leftarrow L_{ij} \\
5: & \hspace{1em} \text{end for} \\
6: & \hspace{1em} t_j^2 \leftarrow \max\{t_i^2 + t_{ij} \mid i \in \text{Prec}(j)\}; \text{label}(j) \leftarrow false \\
7: & \hspace{1em} \text{end for} \\
8: & \text{for all } i \in \text{Prec}(l) \text{ do} \\
9: & \hspace{1em} \text{if } i \neq k \text{ then} \\
10: & \hspace{2em} t_i^l \leftarrow L_i \\
11: & \hspace{1em} \text{end if} \\
12: & \text{end for} \\
13: & t_{kl} \leftarrow T_{kl}; \\
14: & t_l^2 \leftarrow \max\{t_i^2 + t_{il} \mid i \in \text{Prec}(l)\}; \text{label}(l) \leftarrow false; \\
15: & \text{if } t_l^2 \neq t_k^2 + t_{kl} \text{ then} \\
16: & \hspace{1em} PossCritical \leftarrow false; \text{exit} \\
17: & \text{end if} \\
\triangleright \text{PHASE 2:} \\
18: & \text{label}(l) \leftarrow true; \\
19: & \text{for } j \leftarrow l + 1 \text{ to } n \text{ do} \\
20: & \hspace{1em} \text{for all } i \in \text{Prec}(j) \text{ do} \\
21: & \hspace{2em} \text{if } \text{label}(i) = true \text{ then} \\
22: & \hspace{3em} t_{ij} \leftarrow T_{ij} \\
23: & \hspace{2em} \text{else} \\
24: & \hspace{3em} t_{ij} \leftarrow L_{ij} \\
25: & \hspace{2em} \text{end if} \\
26: & \hspace{1em} \text{end for} \\
27: & t_j^2 \leftarrow \max\{t_i^2 + t_{ij} \mid i \in \text{Prec}(j)\}; \\
28: & \text{if } \{i \mid i \in \text{Prec}(j), \text{label}(i) = true, t_j^2 = t_i^2 + t_{ij}\} \neq \emptyset \text{ then} \\
29: & \hspace{1em} \text{label}(j) \leftarrow true \\
30: & \text{else} \\
31: & \hspace{1em} \text{label}(j) \leftarrow false \\
32: & \hspace{1em} \text{end if} \\
33: & \text{end for} \\
34: & PossCritical \leftarrow \text{label}(n); 
\end{align*}
Algorithm 1, rows: 27, 28, 31). Hence there exists a path containing a path from node critical and therefore it is possibly critical. This contradicts the assumption that it is possibly critical in \( T' \). Thus we arrive to contradiction, since \( p^* \) traversing \((k, l)\) is not critical in \( T' \).

\[ (\Leftarrow) \) Assume that Algorithm 1 returns \( \text{PossCritical} = \text{true} \) for \((k, l)\). Then, \( \text{label}(l) = \text{true} \) and \( \text{label}(n) = \text{true} \). This implies that in a configuration \( T \in \mathcal{S}^*(\bar{T}_{kl}) \) returned by the algorithm all longest paths from 1 to \( l \) use activity \((k, l)\) and there exists a path from \( l \) to \( n \) whose nodes are labeled \text{true} and therefore they are critical (see Algorithm 1, rows: 27, 28, 29). Hence there exists a path containing \((k, l)\), critical in \( T \). By Definition 1, it is possibly critical and by Proposition 1 \((k, l)\) is possibly critical.

It is clear that the running time of Algorithm 1 is \( \mathcal{O}(m) \).

Remark 1. The algorithm distinguishing whether an activity \((k, l)\) is possibly critical in \( G \) with activities \((i,j) \in \text{SUCC}(k, l)\) having precise duration times, can be identical to Algorithm 1. It is enough to reverse arcs in network \( G \) and carry out the computations from node \( n \) down to 1.

2.1.2 A polynomial algorithm for determining the lower bound on the latest starting time of an activity

We now present an algorithm for solving PLBLST. Let us recall an important result, identical to Algorithm 1. It is enough to reverse arcs in network \( G \).

Proposition 3. \( t^l_{kl} = \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} t^l_{kl}(T) \). Moreover, the minimum \( t^l_{kl}(T) \) is attained on the vertices of the hyper-rectangle \( \mathcal{S}^*(\bar{T}_{kl}) \).

The key lemma for constructing the algorithm for computing \( t^l_{kl} \) (Algorithm 2) is the following one.

Lemma 3. Let \( f^*_{kl} \) be the minimal nonnegative real number such that \((k, l)\) with a duration time \( \bar{t}_{kl} + f^*_{kl} \) becomes possibly critical. Then \( t^f_k + f^*_k = \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} t^l_{kl}(T) \), where \( t^f_k \) is the earliest moment when event \( k \) occurs.

Proof. Let us observe that \( t^l_{kl}(T) = t^f_k + f^*_k \) for all \( T \in \mathcal{S}^*(\bar{T}_{kl}) \). This follows from the fact that \( t^f_k(T) \) is equal to \( f^*_k \) for all \( T \in \mathcal{S}^*(\bar{T}_{kl}) \). Thus to prove \( t^f_k + f^*_k = \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} t^l_{kl}(T) \), we only need to show that \( f^*_k = \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} f^l_{kl}(T) \). Assume to the contrary that \( f^*_k \neq \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} f^l_{kl}(T) \). Let us consider two cases.

Case 1: \( f^*_k < \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} f^l_{kl}(T) \). The result is \( \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} f^l_{kl}(T) > 0 \) and consequently \((k, l)\) with duration time \( \bar{t}_{kl} + f^*_k \) is not critical for all \( T \in \mathcal{S}^*(\bar{T}_{kl} + f^*_k) \). This contradicts the possible criticality of \((k, l)\).

Case 2: \( f^*_k > \min_{T \in \mathcal{S}^*(\bar{T}_{kl})} f^l_{kl}(T) \). Hence there exists a configuration \( T' \in \mathcal{S}^*(\bar{T}_{kl}) \) such that \( f^*_k > f^l_{kl}(T') \). Let us increase the duration time of \((k, l)\) from \( \bar{t}_{kl} + f^l_{kl}(T') \) in \( T' \). For this new configuration, say \( T'' \in \mathcal{S}^*(\bar{T}_{kl} + f^l_{kl}(T')) \), \((k, l)\) is critical and therefore it is possibly critical. This contradicts the assumption that \( f^*_k \) is the minimal number such that \((k, l)\) with duration time \( \bar{t}_{kl} + f^*_k \) becomes possibly critical.
Algorithm 2 Computing the minimal latest starting time of an activity

Require: A network \( G = \langle V, A \rangle \), a specified activity \((k, l) \in A\), time intervals \( T_{ij} = [t_{ij}, t_{ij}]\), \((i, j) \in A\)

Ensure: The minimal latest starting time of \((k, l)\), \( t_{kl} \).

1: \( f_{kl} \leftarrow 0; \)
   \( \text{// Check possible criticality of \((k, l)\).} \)
2: call Algorithm 1;
3: \( \text{while not PossCritical do} \)
4: \( \text{if label}(l) \text{ then} \)
5: \( \Delta \leftarrow \min\{t_i^e - t_i - t_{ij} \mid (i, j) \in A, \text{label}(i) = \text{true}, \text{label}(j) = \text{false}\} \)
6: \( \text{else} \)
7: \( \Delta \leftarrow t_i^e - t_i^e - t_{kl} \)
8: \( \text{end if} \)
9: \( t_i^e \leftarrow t_i^e + \Delta; \)
10: \( f_{kl} \leftarrow f_{kl} + \Delta; \)
   \( \text{// Check possible criticality of \((k, l)\) with implicitly increased duration} \)
   \( \text{// \( t_{kl} + f_{kl} \).} \)
11: call only PHASE 2 of Algorithm 1
12: \( \text{end while} \)
13: \( f_{kl} \leftarrow f_{kl}^e; \) \( \text{// \( f_{kl} \) equals} f_{kl}^e \)

The idea of Algorithm 2 is based on Lemma 3. It consists in finding the minimal nonnegative real number that added to the upper bound of duration interval of a specified \((k, l)\) makes it possibly critical. In each iteration of the algorithm the duration time of \((k, l)\) is increased (row 9) and the possible criticality of \((k, l)\) for such an increased duration is tested. The testing is reduced to applying the algorithm for asserting the possible criticality of \((k, l)\) (Algorithm 1). This process is repeated until the activity becomes possibly critical. Then from Lemma 3, we immediately obtain the minimal latest starting time of \((k, l)\). It is worth noticing that in row 11 only PHASE 2 of Algorithm 1 is called, because the earliest moments of the occurrence of events \( t_i^e \), node labels \( \text{label}(i) \), and duration times \( t_{ij} \), for \( i, j \leq l \), computed in the first call (row 2) remain unchanged.

Figure 1: A network with duration intervals.

To clarify Algorithm 2, we apply it to the network in Figure 1. Activity \((2, 3)\) is distinguished. The network after the first, second and third call of Algorithm 1 is given in Figures 2, 3 and 4, respectively. Nodes labeled true after the calls of Algorithm 1 are black, activities whose free floats are computed in row 5 (see Algorithm 2) are
Figure 2: The network, given in Figure 1, after the first call of Algorithm 1 in Algorithm 2.

Figure 3: The network, given in Figure 1, after the second call of Algorithm 1 - PHASE 2 - in Algorithm 2.

marked with a dashed line, activities whose duration times are at their upper bounds are shown in bold print. A possibly critical path, determined by Algorithm 1, that traverses activity (2, 3) is presented in Figure 4, $1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 7$. Thus (2, 3) is possibly critical. The number $f_{23} = 2$, in Figure 4, is the minimal one such that when added to $T_{23}$, it makes (2, 3) possibly critical. From this and Lemma 3, we obtain the minimal latest starting time $t_{23} = 5$.

**Lemma 4.** Algorithm 2 finds $t_{kl}$.

**Proof.** To prove that a quantity $t_k^l + f_{kl}$, computed by Algorithm 2 (row 13), is the minimal latest starting time of activity $(k, l)$, we only need to show, by Lemma 3, that $f_{kl}$ is the minimal nonnegative real number such that $(k, l)$ with duration time equal to $t_{kl} + f_{kl}$ becomes possibly critical. Then making use of Lemma 3, we immediately obtain the minimal latest starting time of $(k, l)$.

We use a proof by contradiction. Suppose that there exists a nonnegative real number $f'_{kl}$ such that

$$f'_{kl} < f_{kl},$$

and $(k, l)$ becomes possibly critical for a duration time equal to $t_{kl} + f'_{kl}$. We denote the number of calls of Algorithm 1 in Algorithm 2 by $q$. This implies that $f_{kl} = \sum_{i=1}^{q-1} \Delta_i$ (see Algorithm 1, row 10). Note that for the duration time of $(k, l)$ equal to $t_{kl} + \sum_{i=1}^{q-2} \Delta_i$, Algorithm 1 asserts that $(k, l)$ is not possibly critical. Additionally, it returns $\text{PossCritical} = false$, a configuration of duration times $T', T' \in \mathcal{S}(T_{kl} +$
Lemma 5. The running time of Algorithm 2 is $\mathcal{O}(mn)$.

Proof. Algorithm 1 ($\mathcal{O}(m)$) is invoked in Algorithm 2 at most $n-2$ times (until $\text{label}(n) = \text{true}$). In each call of Algorithm 1 at least one node is labeled $\text{true}$ that in previous calls was labeled $\text{false}$ (nodes labeled $\text{true}$ in previous calls have unchanged labels in consecutive calls). Hence the entire complexity of Algorithm 2 is $\mathcal{O}(mn)$.

Figure 4: The network, given in Figure 1, after the third call of Algorithm 1 - PHASE 2 - in Algorithm 2.
2.2 Determination of the maximal latest starting time of an activity

We now pass on to the problem of computing an upper bound on the latest starting times of an activity. We first examine a problem closely related to it: that of evaluating the necessary criticality of an activity.

2.2.1 Problem of evaluating the necessary criticality of an activity – a polynomially solvable case

Let us recall the notions of the necessary criticality of activities and paths in the network $G$.

**Definition 2.** An activity $(k, l) \in A$ (resp. a path $p \in P$) is necessarily critical in $G$ if and only if for every configuration of times $T \in \mathfrak{T}$, $(k, l)$ (resp. $p$) is critical in $G$ in the usual sense, after replacing the time intervals $T_{ij}$ by exact values $t_{ij}(T)$, $(i, j) \in A$.

The notions of the necessary criticality, both with respect to paths and activities, have been analyzed in [3]. The problem of the necessary criticality for a path can be solved in polynomial time. Unfortunately, the one for an activity does not seem to be such. The question of proving this fact is still open. In [11] a polynomial algorithm has been provided only for series-parallel networks. In the remainder of this section, we assume that activities $(i, j) \in PREC(k, l)$ in $G$ have duration times such that $t_{ij} = \overline{t}_{ij}$. We show that in this special case the problem of evaluating necessary criticality of activity $(k, l)$ is polynomially solvable. For such a network, we define the set of possible configurations of activity durations and denote it by $\mathfrak{T}_{ij}^s(x)$, $x \geq 0$. $\mathfrak{T}_{ij}^s(x)$ is the Cartesian product of $\mathfrak{T}_{ij}^s(t_{kl})$, $(i, j) \in A$, given as follows:

$$\mathfrak{T}_{ij}^s(x) = \begin{cases} [\overline{t}_{ij}, \overline{t}_{ij}] & \text{for } (i, j) \in PREC(k, l), \\ [x, x] & \text{for } (i, j) = (k, l), \\ [\overline{t}_{ij}, \overline{t}_{ij}] & \text{otherwise.} \end{cases} \quad (6)$$

In the same manner, we can define the set of possible configurations of activity durations $\mathfrak{T}_{ij}^p(x)$, $x \geq 0$, if activities $(i, j) \in SUCC(k, l)$ have precise durations. $\mathfrak{T}_{ij}^p(x)$ is the Cartesian product of $\mathfrak{T}_{ij}^p(t_{kl})$, $(i, j) \in A$, given as follows:

$$\mathfrak{T}_{ij}^p(x) = \begin{cases} [\overline{t}_{ij}, \overline{t}_{ij}] & \text{for } (i, j) \in PREC(k, l), \\ [x, x] & \text{for } (i, j) = (k, l), \\ [\overline{t}_{ij}, \overline{t}_{ij}] & \text{otherwise.} \end{cases} \quad (7)$$

**Proposition 4.** Let activities $(i, j) \in PREC(k, l)$ (resp. $(i, j) \in SUCC(k, l)$) have precise duration times. Activity $(k, l)$ is not necessarily critical in $G$ if and only if there exists a configuration $T \in \mathfrak{T}^s(t_{kl})$ (resp. $T \in \mathfrak{T}^p(t_{kl})$) such that none of critical paths, in $T$, uses $(k, l)$.

**Proof.** Obvious. It results from Definition 2, Propositions 6, 7 and the assumption that activities $(i, j) \in PREC(k, l)$ (resp. $(i, j) \in SUCC(k, l)$) have precise duration times.

Thus, the problem of evaluating the necessary criticality of $(k, l)$ in $G$, in which the durations of activities $(i, j) \in PREC(k, l)$ (resp. $(i, j) \in SUCC(k, l)$) are precisely
We denote the configuration of the activity duration times after the termination by \( \mathcal{G} \).

Algorithm 3 enables us to assert whether an activity \((k, l)\) is necessarily critical in \( G \), under the assumption that activities \((i, j)\) have precise duration times. The main idea of an algorithm (Algorithm 3) which can evaluate the necessary criticality of activity \((k, l)\) is to find a configuration \( T \in \mathcal{T} \) in which \((k, l)\) is not critical in the usual sense. If such a configuration \( T \) is successfully determined then \((k, l)\) is not necessarily critical, otherwise it is. The algorithm is similar in spirit to Algorithm 1. The approach here applied is complementary to the one in Algorithm 1.

To find a configuration \( T \), node labeling is performed with a convenient setting of the activity durations. A node \( i \in V \) is labeled true if the longest path from node 1 to \( i \) uses activities parallel to \((k, l)\), and it is longer (strictly) than all paths from 1 to \( i \) traversing \((k, l)\), in a systematically constructed configuration \( T \), and false otherwise.

The duration time of activity \((i, j)\) is set to its lower bound. We will illustrate the algorithm with an example in Section 2.2.2.

The correctness of Algorithm 3 follows from the following lemma.

**Lemma 6.** Let activities \((i, j)\) have precise duration times. Activity \((k, l)\) is not necessarily critical in \( G \) if and only if Algorithm 3 returns \( \text{NecCritical} = \text{false} \) for \((k, l)\).

**Proof.** \((\Rightarrow)\) Suppose the assertion of the lemma is false, i.e. \( \text{NecCritical} = \text{true} \). We denote the configuration of the activity duration times after the termination by \( T' \).

By assumption, activity \((k, l)\) is not necessarily critical in \( G \) in which activities \((i, j)\) have precise duration times, hence there exists a configuration \( T^* \in \mathcal{T} \) such that none of the critical paths uses \((k, l)\) in \( T^* \) (see Proposition 4). Consequently there exists a critical path \( p^* \), that traverses activities parallel to \((k, l)\), \( p^* \notin P(k, l) \), with the length \( l_{p^*}(T^*) \) satisfying the following inequality

\[
l_{p^*}(T^*) > l_p(T^*), \text{ for all } p \in P(k, l). \tag{8}
\]

Moreover, \( p^* \) is possibly critical and therefore it is critical in the configuration \( T'' \) determined as in Lemma 1. It easy to check that inequality (8) holds in \( T'' \).

Now becomes

\[
l_{p^*}(T'') > l_p(T''), \text{ for all } p \in P(k, l). \tag{9}
\]

Note that (9) states that \((k, l)\) is not necessarily critical.

Let us choose a subpath of \( p^* \) from 1 to \( r \), we denote it by \( p_{1r}^* \), whose nodes \( i \) are labeled true except for end node \( r \). \( p_{1r}^* \) exists, since \( p \in P \) and \( \text{NecCritical} = \text{true} \) and in consequence \( \text{label}(n) = \text{false} \) (see Algorithm 3, row 34). Hence duration times \( t_{ij}(T') \) of activities \((i, j)\) are equal to their upper bounds (see Algorithm 3, rows: 4, 22). This forces \( t_{ij}(T') = t_{ij}(T'') \) for all \((i, j)\) \( p_{1r}^* \). The same equality holds for all \((i, j) \notin \text{SUCC}(k, l) \), since \( T', T'' \in \mathcal{T} \). Since \( \text{label}(r) = \text{false} \), there exists a path \( p_{1r}^* \) from 1 to \( r \) that uses \((k, l)\) with duration times in \( T' \) at their lower bounds and of the length greater or equal to the length \( p_{1r}^* \) in \( T' \) (see Algorithm 3,
Algorithm 3 Asserting whether an activity is necessarily critical

Require: A network $G = (V, A)$, a specified activity $(k, l) \in A$, time intervals $T_{ij} = [L_{ij}, U_{ij}]$, $(i, j) \in A$.

Ensure: A configuration $T \in \mathcal{T}(T_{kl})$, $\text{NecCritical} = \text{true}$ if $(k, l)$ is necessarily critical false otherwise.

\textbf{PHASE 1:}
1: $t^*_k \leftarrow 0; \text{label}(1) \leftarrow \text{true};$
2: for $j \leftarrow 2$ to $l - 1$ do
3: for all $i \in \prec(j)$ do
4: $t_{ij} \leftarrow \overline{t}_{ij}$
5: end for
6: $t^*_j \leftarrow \max\{t^*_i + t_{ij} \mid i \in \prec(j)\}; \text{label}(j) \leftarrow \text{true}$
7: end for
8: for all $i \in \prec(l)$ do
9: if $i \neq k$ then
10: $t_l \leftarrow \overline{t}_l$
11: end if
12: end for
13: $t_{kl} \leftarrow t_{kl};$
14: $t^*_l \leftarrow \max\{t^*_i + t_{il} \mid i \in \prec(l)\}; \text{label}(l) \leftarrow \text{true};$
15: if $t^*_l \neq t^*_k + t_{kl}$ then
16: $\text{NecCritical} \leftarrow \text{false}; \text{exit}$
17: end if

\textbf{PHASE 2:}
18: $\text{label}(l) \leftarrow \text{false};$
19: for $j \leftarrow l + 1$ to $n$ do
20: for all $i \in \prec(j)$ do
21: if $\text{label}(i) = \text{true}$ then
22: $t_{ij} \leftarrow \overline{t}_{ij}$
23: else
24: $t_{ij} \leftarrow L_{ij}$
25: end if
26: end for
27: $t^*_j \leftarrow \max\{t^*_i + t_{ij} \mid i \in \prec(j)\}$;
28: if $\{i \mid i \in \prec(j), \text{label}(i) = \text{false}, t^*_j = t^*_i + t_{ij}\} \neq \emptyset$ then
29: $\text{label}(j) \leftarrow \text{false}$
30: else
31: $\text{label}(j) \leftarrow \text{true}$
32: end if
33: end for
34: $\text{NecCritical} \leftarrow \text{not label}(n);$
...concatenate \( p \) with the subpath of \( p^* \) from \( r \) to \( n \) then we obtain the path \( p' \in P(k, l) \) with the length \( l_{p'}(T') \leq l_{p'}(T) \). It is easily seen that this inequality holds for configuration \( T'' \), contrary to (9).

(\( \Leftarrow \)) Assume that Algorithm 3 returns \( \text{NecCritical} = \text{false} \). Then \( \text{label}(l) = \text{true} \) or \( \text{label}(n) = \text{true} \). From this, it may be concluded that in a configuration \( T \in T_\infty^k(T_{kl}) \) returned by the algorithm none of the critical paths uses \((k, l)\) (see Algorithm 3, rows: 14, 15, 16, 27, 28, 31). By Proposition 4, \((k, l)\) is not necessarily critical. This completes the proof.

It is evident that the running time of Algorithm 3 is \( O(m) \).

**Remark 2.** The algorithm asserting whether an activity \((k, l)\) is necessarily critical in \( G \) with activities \((i, j)\) \( \in \text{SUCC}(k, l) \) having precise duration times can be identical to Algorithm 3. It is sufficient to reverse arcs in network \( G \) and carry out the computations from node \( n \) down to 1.

### 2.2.2 A polynomial algorithm for determining the upper bound on the latest starting time of an activity

Here, we propose an algorithm for solving PUBLST.

Let us recall an important result, given by Dubois et al. [9], that allows to reduce the set of configurations \( T \).

**Proposition 5.** \( t^l_{kl} = \max_{T \in T_\infty^k(T_{kl})} t^l_{kl}(T) \). Moreover, the maximum \( t^l_{kl}(T) \) is attained on the vertices of the hyper-rectangle \( T_\infty^k(T_{kl}) \).

**Algorithm 4** Computing the maximal latest starting time of an activity

**Require:** A network \( G = (V, A) \), a specified activity \((k, l) \in A\), time intervals \( T_{ij} = [t_{ij}; t_{ij}'] \), \((i, j) \in A\).

**Ensure:** The maximal latest starting time of \((k, l)\), \( t^l_{kl} \).

1. \( f_{kl} \leftarrow 0; \)
2. \( \triangleright \text{Check necessary criticality of} \,(k, l).\)
3. call Algorithm 3;
4. \( \textit{while not} \ \text{NecCritical} \ \textit{do} \)
5. \( \quad \text{if} \ \text{label}(l) \ \text{then} \)
6. \( \quad \quad \Delta \leftarrow t^l_i - t^l_k - t_{kl}; \)
7. \( \quad \text{else} \)
8. \( \quad \quad \Delta \leftarrow \min\{t^r_j - t^r_i - t_{ij} \mid (i, j) \in A, \text{label}(i) = \text{false}, \text{label}(j) = \text{true}\}; \)
9. \( \quad \text{end if} \)
10. \( t^l_i \leftarrow t^l_i + \Delta; \)
11. \( f_{kl} \leftarrow f_{kl} + \Delta; \)
12. \( \triangleright \text{Check necessary criticality of} \,(k, l)\ \text{with implicitly increased duration} \ t^r_k + f_{kl}. \)
13. \( \text{call only PHASE 2 of Algorithm 3} \)
14. \( \textit{end while} \)
15. \( t^l_{kl} \leftarrow t^l_k + f_{kl}; \ \triangleright \ f_{kl} \ \text{equals} \ f^l_{kl} \)

The construction of the algorithm for determining \( t^l_{kl} \) of a given activity \((k, l) \in A\) (Algorithm 4) is based on the following lemma.
Lemma 7. Let $f_{kl}^*$ be the minimal nonnegative real number such that $(k, l)$ with a duration time $T_{kl} + f_{kl}^*$ becomes necessarily critical. Then $T_k^* + f_{kl}^* = \max_{T \in \mathcal{E}_k (l_{kl})} t_{kl}(T)$, where $T_k^*$ is the earliest moment when event $k$ occurs.

Proof. Our proof starts with the observation that $t_{kl}(T) = T_k^* + f_{kl}(T)$, for all $T \in \mathcal{E}_k (l_{kl})$. The observation follows from the fact that $t_{kl}^*(T) = T_k^*$, for all $T \in \mathcal{E}_k (l_{kl})$.

Hence in order to prove $T_k^* + f_{kl}^* = \max_{T \in \mathcal{E}_k (l_{kl})} t_{kl}(T)$, it suffices to show that $f_{kl}^* = \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T)$. Suppose on the contrary that $f_{kl}^* \neq \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T)$. Then we should consider the following two cases.

Case 1: $f_{kl}^* < \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T)$. This implies that there exists a configuration $T' \in \mathcal{E}_k (l_{kl})$ such that $f_{kl}^* < f_{kl}(T')$, which gives $f_{kl}(T') > 0$. Consequently $(k, l)$ is not critical in $T'$. Let us increase the duration time of $(k, l)$ from $T_k$ to $T_k + f_{kl}^*$ in $T'$. For this new configuration, say $T'' \in \mathcal{E}_k (l_{kl} + f_{kl}^*)$, $(k, l)$ is still not critical, which contradicts the assumption that $(k, l)$ is critical for all $T \in \mathcal{E}_k (l_{kl} + f_{kl}^*)$.

Case 2: $f_{kl}^* > \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T)$. Thus $(k, l)$ is critical with duration time $T_{kl} + \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T)$, for all $T \in \mathcal{E}_k (l_{kl} + \max_{T \in \mathcal{E}_k (l_{kl})} f_{kl}(T))$, and therefore it is necessarily critical. This contradicts the assumption that $f_{kl}^*$ is the minimal number such that $(k, l)$ becomes necessarily critical.

Algorithm 4 enables to solve PUBLST. The main idea of the algorithm is based on Lemma 7. It consists in determining the minimal nonnegative real number that added to the lower bound of the duration interval of a specified activity $(k, l)$ makes it the necessarily critical. Namely, in each iteration the duration time of $(k, l)$ is suitably increased (row 9) and necessary criticality of $(k, l)$ for such an increased duration is evaluated. The evaluation boils down to applying Algorithm 3. This process is repeated until the activity becomes necessarily critical. Then Lemma 7 gives the maximal latest starting time of $(k, l)$. It is worth noticing that in row 11 only Phase 2 of Algorithm 3 is called, because the earliest moments of the occurrence of events $t_{ij}^*$, node labels $\text{label}(i)$, and duration times $t_{ij}$, for $i, j \leq l$, computed in the first call (row 2) remain unchanged.

![Figure 5](image-url)

Figure 5: The network, given in Figure 1, after the first call of Algorithm 3 in Algorithm 4.

We now clarify Algorithm 4 by an illustrative example. Consider the network in Figure 1. Activity $(2, 3)$ is distinguished. The network after the first, second and third call of Algorithm 3 is given in Figures 5, 6 and 7, respectively. Nodes labeled $\text{true}$ after the calls of Algorithm 3 are black, activities whose free floats are computed in row
Figure 6: The network, given in Figure 1, after the second call of Algorithm 3 - Phase 2 - in Algorithm 4.

Figure 7: The network, given in Figure 1, after the third call of Algorithm 3 - Phase 2 - in Algorithm 4.

7 (see Algorithm 4) are marked with a dashed line, activities whose duration times are at their upper bounds are shown in bold print. Activity (2, 3) remains not necessarily critical with the duration times equal to 1 and 6. For both, Algorithm 3 successfully determined the configurations in which there exists the critical path, $1 \rightarrow 6 \rightarrow 7$, that uses activities parallel to (2, 3) with the length greater than the lengths of all paths crossing (2, 3) (see Figures 5, 6). (2, 3) becomes necessarily critical with the duration time equal to 13, since the algorithm could not find such a configuration (see Figure 7). The number $f_{23} = 12$, in Figure 7, is the minimal one such that added to $t_{23}$, makes (2, 3) necessarily critical. From this and Lemma 7, we obtain the maximal latest starting time $t_{l23} = 16$.

**Lemma 8.** Algorithm 4 finds $\bar{t}_{kl}$.

**Proof.** The proof goes in the similar manner to the one of Lemma 4. Namely, it suffices to show that $f_{kl}$, computed by Algorithm 4 (row 13), is the minimal nonnegative real number such that $(k, l)$ with duration time equal to $t_{kl} + f_{kl}$ becomes necessarily critical. Then, making use of Lemma 7, we immediately obtain the maximal latest starting time $(k, l)$.

Suppose, by contradiction, that there exists a nonnegative real number $f'_{kl}$ such that $(k, l)$ becomes necessarily critical with $t_{kl} + f'_{kl}$ and satisfying the following inequality

$$f'_{kl} < f_{kl}. \quad (10)$$
We denote the number of calls of Algorithm 3 in Algorithm 4 by \( q \). This clearly forces \( f_{kl} = \sum_{i=1}^{q-1} \Delta_i \) (see Algorithm 4, row 10). For the duration time of \((k,l)\) equal to \( t_{kl} + \sum_{i=1}^{q-2} \Delta_i \), Algorithm 3 asserts that \((k,l)\) is not necessarily critical. Furthermore, it returns \( \text{NecCritical} = \text{false} \), a configuration of duration times \( T', T'' \in \mathbb{T} (t_{kl} + \sum_{i=1}^{q-2} \Delta_i) \), node labels, \( \text{label}_i(T'), i = 1, \ldots, n \). For the duration time of \((k,l)\) equal to \( t_{kl} + f_{kl}' \), Algorithm 3 asserts the necessary criticality of \((k,l)\) and returns \( \text{NecCritical} = \text{true} \), a configuration of duration times \( T'', T'' \in \mathbb{T} (t_{kl} + f_{kl}') \), node labels \( \text{label}_i(T'') \), \( i = 1, \ldots, n \). Since \((k,l)\) is necessary critical for \( t_{kl} + f_{kl}' \), but for \( t_{kl} + \sum_{i=1}^{q-1} \Delta_i \) \((k,l)\) is not, we see \( f_{kl}' > \sum_{i=1}^{q-2} \Delta_i \). Thus, \( f_{kl}' = \sum_{i=1}^{q-2} \Delta_i + \delta \) \((\delta > 0)\). From the way of determining configurations \( T' \) and \( T'' \) by Algorithm 3 for \( t_{kl} + \sum_{i=1}^{q-1} \Delta_i \) and \( t_{kl} + \sum_{i=1}^{q-2} \Delta_i + \delta \), respectively, we conclude that there exists at least one activity \((i',j') \in A\) such that

\[
\text{label}_j(T') = \text{false}, \quad \text{label}_j(T') = \text{true},
\]

\[
\text{label}_j(T'') = \text{false}, \quad \text{label}_j(T'') = \text{false}
\]

or

\[
\text{label}_j(T') = \text{true}, \quad \text{label}_j(T'') = \text{false} \quad \text{for} \quad (i',j') = (k,l)
\]

satisfying

\[
t_{i'}(T') - t_{i'}(T') - t_{i'}(T') \leq \delta. \tag{11}
\]

We now consider the stage after \((q - 2)\)th call of Algorithm 3. Algorithm 4 (rows 5 or 7) computes \( \Delta_{q-1} = \min \{ t_{i'}(T') - t_{i'}(T') - t_{ij}(T') \mid (i,j) \neq (k,l), \text{label}_i(T') = \text{false}, \text{label}_j(T') = \text{true} \} \). From this and (11), it follows that \( \Delta_{q-1} \leq \delta \). Consequently \( \sum_{i=1}^{q-2} \Delta_i + \Delta_{q-1} \leq \sum_{i=1}^{q-2} \Delta_i + \delta \) and therefore \( f_{kl} \leq f_{kl}' \), contrary to (10).

**Lemma 9.** The running time of Algorithm 4 is \( \mathcal{O}(mn) \).

**Proof.** The proof proceeds in the same manner as the one of Lemma 5. Algorithm 3 \( (\mathcal{O}(m)) \) is invoked in Algorithm 4 at most \( n - 2 \) times (until \( \text{label}(n) = \text{false} \)). In each call of Algorithm 3 at least one node is labeled \( \text{false} \). Thus the complexity of Algorithm 4 is \( \mathcal{O}(mn) \).

**2.3 Determination of bounds on float of an activity**

This section deals with the problem of determining the bounds on float of a given activity. That is, we consider the problems PLBF and PUBF, announced at the beginning of Section 2. We prove that one cannot approximate PLBF within a factor smaller than 1. Successively, we show, taking into account results obtained in this paper, that PLBF and PUBF are polynomially solvable in some special cases.

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

**Proposition 6.** An activity \((k,l) \in A\) is possibly (resp. necessarily) critical in \( G \) if and only if \( \sum_{k} = 0 \) (resp. \( \sum_{k} = 0 \)).
From the above proposition, it may be concluded that PLBF is at least as hard, if not harder than the problem the possible criticality of an activity and, similarly, PUBF is at least as hard as the problem the necessary criticality of an activity. Accordingly, PLBF is strongly $NP$-hard for general networks and remains $NP$-hard even if the network $G$ is restricted to be planar and regular of degree three (see [7]). Unfortunately, a question still unanswered is whether PUBF is polynomially solvable for general and planar networks. In [9] some heuristic methods for PLBF and PUBF have been proposed. At present, these problems are effectively solvable for networks having a special topology, namely the one of series-parallel graphs, for which the provided algorithms are polynomial [11].

We define the set of possible configurations of activity durations $T(x)$ (resp. $\overline{T}(x)$), $x \geq 0$, to be the Cartesian product of corresponding time intervals $T_{ij}(x)$, $(i, j) \in A$, given as follows:

$$T_{ij}(x) = \begin{cases} [x, x] & \text{for } (i, j) = (k, l), \\ [L_{ij}, L_{ij}] & \text{for } (i, j) \in PREC(k, l) \cup SUCC(k, l), \\ [x, x] & \text{otherwise} \end{cases}$$

The following proposition due to Dubois et al. [9].

**Proposition 7.** $f_{kl} = \min_{T \in \Sigma(l_{kl})} f_{kl}(T)$ (resp. $\overline{f}_{kl} = \max_{T \in \overline{T}(l_{kl})} f_{kl}(T)$) Moreover, the minimum (resp. the maximum) $f_{kl}(T)$ is attained on the vertices of the hyper-rectangle $\Sigma(l_{kl})$ (resp. $\overline{T}(l_{kl})$).

It is worth pointing out that the reduction of the set of the possible configurations $\Sigma$ to $\Sigma(l_{kl})$ and to $\overline{T}(l_{kl})$ (Proposition 7) is not sufficient to keep PLBF from being $NP$-complete and it does not allow to answer the question about the complexity of PUBF.

### 2.3.1 Inapproximability

The negative result for PLBL encourages us to look for approximation algorithms for this problem. Unfortunately, one cannot approximate PLBF within a factor smaller than 1.

Given a minimization problem $P$, let $opt_P(I)$ denote the optimal solution value for some instance $I$ of $P$ and for a solution $y$ of $I$, let $val_P(I, y)$ denote the associated value. Given a constant $\epsilon \in (0, 1)$, a $\epsilon$-approximation algorithm for $P$ is an algorithm that applied to any instance $I$ of $P$, runs in time bounded by polynomial in the size of $I$ and produces a solution $y$ whose value fulfills inequality $val_P(I, y) \leq \frac{1}{1-\epsilon} opt_P(I)$. If such an algorithm exists, we say that $P$ is approximable within $\epsilon$ (see for instance [1]).

A negative approximation result is stated in the following theorem.

**Theorem 1.** If $P \notin NP$ then it is not possible to approximate PLBF within $\epsilon \in (0, 1)$, even when $G$ is restricted to a planar network.

**Proof.** See Appendix.
2.3.2 Some solvable special cases

We now study special cases of PLBF and PUBF for which polynomial algorithms for general networks exist.

Our basic assumption is that activities from set \( PREC(k,l) \) have precise duration times in network \( G \), \( t_{ij} = \tau_{ij} \) for all \((i,j) \in PREC(k,l)\), where \((k,l) \in A\) is a specified.

**Lemma 10.** Assume that activities \((i,j) \in PREC(k,l)\) have precise duration times and \( f_{kl}^* \) is the minimal nonnegative real number such that \((k,l)\) with a duration time equal to \( \tau_{kl} + f_{kl}^* \) (resp. \( \tau_{kl} + f_{kl}^* \)) becomes possibly (resp. necessary) critical. Then \( f_{kl} = f_{kl}^* \) (resp. \( f_{kl} = f_{kl}^* \)).

**Proof.** From the assumption that activities \((i,j) \in PREC(k,l)\) have precise duration times, we get \( \tau(x) = \sum_s(x) \) (resp. \( \tau(x) = \sum_f(x) \)). Thus it suffices to show that \( f_{kl}^* = \min_{T \in \mathbb{E}(\tau_{kl})} \tau_{kl}(T) \) (resp. \( f_{kl}^* = \max_{T \in \mathbb{E}(\tau_{kl})} \tau_{kl}(T) \)). The proof of this equality runs in the same manner as the one of Lemma 3 (resp. Lemma 7).

**Lemma 11.** Assume that activities \((i,j) \in PREC(k,l)\) have precise duration times. Then Algorithm 2 (resp. Algorithm 4) finds \( f_{kl}^* \), \( f_{kl} = \tilde{f}_{kl} - \tilde{f}_k \) (resp. \( \tilde{f}_{kl} \), \( \tilde{f}_{kl} = \tilde{f}_{kl}^* - \tilde{f}_k \)).

**Proof.** Algorithm 2 (resp. Algorithm 4) finds \( t_{kl}^* \) (resp. \( \tau_{kl}^* \)) by Lemma 4 (resp. Lemma 8). That \( t_{kl}^* = t_{k}^* + f_{kl}^* \) (resp. \( \tau_{kl}^* = \tau_{k}^* + f_{kl}^* \)), where \( f_{kl}^* \) is the minimal nonnegative real number such that \((k,l)\) with a duration time equal to \( \tau_{kl} + f_{kl}^* \) (resp. \( t_{kl} + f_{kl}^* \)) becomes possibly (resp. necessary) critical over the set \( \sum_s(\tau_{kl} + f_{kl}^*) \) (resp. \( \sum_f(\tau_{kl} + f_{kl}^*) \)). The proof of this follows from Lemma 3 (resp. Lemma 7). We thus get \( f_{kl}^* = \tilde{t}_{kl} - \tilde{t}_k \) (resp. \( f_{kl}^* = \tilde{\tau}_{kl} - \tilde{\tau}_k \)). Activities \((i,j) \in PREC(k,l)\) have precise duration times, and so \( \tau(x) = \sum_s(x) \) (resp. \( \tau(x) = \sum_f(x) \)). Hence and Lemma 10 we conclude that \( f_{ij}^* = f_{ij}^* \) (resp. \( f_{ij} = f_{ij}^* \)).

If activities \((i,j) \in SUCC(k,l)\) have precise duration times, where \((k,l) \in A\) is a specified, then PLBF and PUBF are also polynomially solvable. It is sufficient to reverse the arcs in network \( G \) and apply Algorithm 2 and Algorithm 4, respectively, with certain modifications in Algorithm 1 and Algorithm 3 (see Remarks 1 and 2) together with Lemma 11.

3 The latest starting times and the floats of activities in a network with fuzzy durations

Now we focus on the fuzzy case. All the elements of the network \( G \) are the same as in the interval case except for the activity duration times, which are determined by means of fuzzy numbers \( \tilde{t}_{ij} \), \((i,j) \in A\), which imprecisely determine duration times of activities \((i,j) \in A\). Fuzzy number \( \tilde{t}_{ij} \) expresses uncertainty connected with the ill-known activity duration time modeled by this number. It generates a possibility distribution for the sets of values containing the unknown activity duration. More formally, we say that the assertion of the form “\( t_{ij} \) is \( \tilde{t}_{ij} \)”, where \( t_{ij} \) is a variable and
In this paper, we have considered the problems of computing the bounds on the latest starting times and the floats of activities in a network with imprecise durations represented by means of interval or fuzzy numbers. We have proposed \(O(mn)\) time algorithms for computing the intervals of the possible values of the latest starting times of an activity in general networks with interval duration times. So far, there have existed polynomial algorithms only for networks having a series-parallel topology. Furthermore, we have proved that the problem of computing the bounds on the floats of an
activity in networks with interval durations cannot be approximated within a factor smaller than 1 and described some polynomially solvable cases.

We have also shown how to use the proposed algorithms for computing the fuzzy latest starting times of activities in a network with activity duration times given in the form of fuzzy numbers.

References


Appendix

The proof of Theorem 1

The proof is inspired by the $\mathcal{NP}$-completeness proof of the problem of the possible criticality of an activity, presented in [7] (Theorem 1). To prove a polynomial reduction from a certain modified PARTITION problem, denoted MPARTITION, to the one of evaluating possible criticality has been applied there.

The MPARTITION is defined as follows:

**INPUT:** A finite set $A$ of positive integers, $A = \{a_1, \ldots, a_q\}$, having the overall sum of $2b$ and a positive integer $K < q$.

**QUESTION:** Is there a subset $A' \subset A$ that sums up exactly to $b$ and $|A'| = K$?

It is well known that MPARTITION is $\mathcal{NP}$-complete (see for instance [13] and comments on PARTITION given there).

We use the proof by contradiction. Suppose that for some $\epsilon < 1$, there is a $\epsilon$-approximation algorithm for PLBF. The idea of the proof is to use the algorithm to construct a polynomial algorithm to solve MPARTITION. Since MPARTITION is $\mathcal{NP}$-complete, we get a contradiction if $\mathcal{P} \neq \mathcal{NP}$.

We now polynomially transform an instance of MPARTITION to an instance of PLBF for a planar network.
The transformation is divided into 2 steps. In the first one we associate to each instance of MPARTITION a directed, acyclic, planar network $G'(A', V')$ with $4q + 3$ nodes (events) labeled $1, 2, \ldots, 4q + 3$. The construction of $G'$ is the same as in [7] (the proof of Theorem 1). In the second step we transform network $G'$ into $G''(A'', V'')$. That is, we multiply the interval bounds by $\frac{1}{1 - \epsilon} M$, where $M$ is a sufficiently large number (for instance $M = q + 1$), and add 1 to the interval bounds of activity $(2q + 2, 4q + 3)$ (see Figure 8). Activities $(2i, 2(2q - i + 2)), i = 1, \ldots, q$, have time intervals $\left[\frac{M}{1 - \epsilon} (q \sum_{j=i+1}^{q} a_j + q - i + 1), \frac{M}{1 - \epsilon} (q \sum_{j=i+1}^{q} a_j + q - i + 1)\right]$, the one-point intervals have been written in Figure 8 as precise times, $(2q + 1, 2q + 3)$ is a specified activity.

The following claim may be proved for $G''$.

**Claim:** If there exists a subset $A' \subset A$ that sums up exactly to $b$ and $|A'| = K$ then the lower bound on the float of activity $(2q + 1, 2q + 3)$ is equal to $1$, i.e., $f_{2q+1,2q+3} = 1$.
in $G'$. Otherwise it is at least $\frac{M}{1-\epsilon}$.

The proof of the claim, by Proposition 6, proceeds in the same manner as the one that MPARTITION is polynomially reducible to the problem of evaluating the possible criticality of activity $(2q + 1, 2q + 3)$ in $G'$ (see the proof of Theorem 1 in [7]).

Consider what happens when we run the $\epsilon$-approximation algorithm for PLBF on $G''$. If there exists a subset $A' \subseteq A$ that sums up exactly to $b$ and $|A'| = K$ then the algorithm returns the float of activity $(2q + 1, 2q + 3)$ whose value is at most $\frac{1}{1-\epsilon}$. Otherwise this value is at least $\frac{M}{1-\epsilon}$. In effect, we have given a polynomial time algorithm for MPARTITION. This contradicts the fact that MPARTITION is $\mathcal{NP}$-complete, unless $\mathcal{P} = \mathcal{NP}$. This completes proof.