Computing efficiently floats of all activities in a series-parallel network with duration intervals*

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 problem of computing intervals of possible values of floats for all activities in a series-parallel network with duration intervals. We present an $\mathcal{O}(n)$ time algorithm, using graph reductions, for computing the interval of possible values of floats for a single activity and an algorithm for computing the intervals for all activities in the network that runs $\mathcal{O}(n \log n)$ time and requires a linear space for a data structure.

Keywords: Graph reductions; Series-parallel graph; Project scheduling; Data intervals

1 Introduction

One of the basic problems in project scheduling is that of the identification of the critical activities, and determining the earliest and latest starting times of activities in an activity network representing a project, where activities with given duration times are related to each other by means of precedence constraints. The identification of *critical activities*, i.e. the activities which, under the assumption of minimum project duration, have no time float for their execution and must be started and completed on strictly determined moments, is carried out via the Critical Path Method [11] (CPM). This method also calculates for each activity, as a by-product, the earliest and latest starting times of activities. The difference between these calculated values is the *float* of activity. An activity is *critical* if and only if its float is equal to zero. Therefore, under the assumption that the durations of activities are precisely known, the problem of determining all these project characteristics is rather simple. When activity duration times are ill-known the problem becomes more complicated even if their estimations are modeled by intervals. The overwhelming part of the literature devoted to this topic adopts an stochastic approach to model uncertain durations and thus leads

*This work was supported by grant no. 7T11F02120 from the State Committee for Scientific Research (Komitet Badań Naukowych).

to intractable problems that are still partially unsolved, except for series parallel networks (see [12] for a bibliography). On the other hand there is a fuzzy approach, i.e. uncertain durations are modeled by fuzzy numbers, The interval approach seems to have existed only as a special case of the fuzzy approach. Since the late 70's several authors have tried to provide methods for determining the earliest and latest starting times of activities and the identification of critical activities in networks with uncertain durations modeled by fuzzy or interval numbers [3],[10],[13],[14] (see [4] for a brief survey and a full bibliography). The proposed methods, mainly based on the CPM, calculate the earliest starting times of activities in correct a way, but they fail to calculate the latest starting times. So, floats can no longer be recovered from earliest and latest starting times and do not lead to the unique identification of critical activities. Recently, Chanas et al. [4], [5] have studied the *possible and necessary criticality* of activities in networks with interval duration times. Dubois et al. [8] have studied these notions of criticality from the point of view of floats. An activity is *necessarily* critical if it is critical whatever the actual values of activity durations turn out to be. An activity is *possibly* critical, when there exist values of durations leading to a configuration of the network where the activity is critical. In [6], it has been proved that the problem of evaluating the possible criticality of an activity is strongly $\mathcal{NP}$-complete for a general network and remains $\mathcal{NP}$-complete even when a network is restricted to be planar (see [7]). A direct consequence of these complexity results is the complexity of the problem of determining the interval (bounds) of possible values of floats for a given activity in networks with interval duration times. Namely, the problem of computing the lower bound of the interval is strongly $\mathcal{NP}$-hard for general networks and remains $\mathcal{NP}$-hard even for planar networks. Unfortunately, the question, still unanswered, is whether the problem of computing the upper bound is polynomially solvable. Both problems have been completely solved by Fargier et al. [9] when a network is series parallel. Fargier et al. have provided $\mathcal{O}(n)$ algorithms for them and an $\mathcal{O}(n)$ algorithm for computing the bounds on latest starting times of an activity for this network. (In [17], there has been given an $\mathcal{O}(mn)$ algorithm for computing the bounds on latest starting times of an activity for a general network).

In this paper we wish to investigate the problem of computing the bounds on floats of all activities in a series-parallel network, where activity duration times are specified as interval numbers. The paper is organized as follows. Section 2 reviews classical graph-theoretic definitions and notations that will be used throughout the paper. Section 3 formally describes the problem that we consider. Section 4.1 gives an $\mathcal{O}(n)$ time algorithm for computing the bounds on floats of a single activity in a series-parallel network. Our algorithm uses series and parallel graph reductions preserving the bounds on floats of a distinguished activity. It works for series-parallel networks and can be used for simplifying the problem of computing the bounds on floats of a single activity in general networks, which is $\mathcal{NP}$-hard. The algorithm can be seen as an alternative to that proposed by Fargier et al. [9]. Both algorithms have the same complexity. Therefore, applying either our algorithm or the one provided by Fargier et al. [9] to each activity in a network for computing the bounds on floats of all activities leads to a method that requires $\mathcal{O}(n^2)$ times. Section 4.2 describes an algorithm for computing the bounds on floats of all activities in series-parallel networks that runs in $\mathcal{O}(n \log n)$ time and requires $\mathcal{O}(n)$ space for a data structure. It takes advantage of dynamic expression trees of Cohen and Tamassia [2], dynamic trees of Sleator and

Tarjan [15] and results proved in [9]. Section 5 presents final remarks.

2 Preliminaries

A directed graph $G =< V, A >$ consists of a finite set V of *nodes*, $|V| = n$, and a finite set A of *arcs* (multiple arcs between the same two nodes are permitted), $|A| = m$. The *in-degree* of node v in G, denoted by $in(v)$, is the number of distinct arcs that enter v, while $out(v)$ denotes *out-degree* of node v in G, the number of arcs that leave v.

A *two-terminal directed acyclic graph* (st-dag for short) is a directed graph without any *cycle* and having a unique source s and a unique sink t. This imply that an st-dag is connected.

When we say an st-dag is *series-parallel* we mean that it is two-terminal arc series-parallel (see [16]). This st-dag is recursively defined as follows (see Figure 1a):

- An st-dag having a single arc is two-terminal series-parallel
- If G_1 and G_2 are two-terminal series-parallel, so are the graphs constructed by each of the following operations:
    1. *Parallel composition:* Identify the source of G_1 with the source of G_2 and the sink of G_1 with the sink of G_2.
    2. *Series composition:* Identify the sink of G_1 with the source of G_2.

Now we recall some properties of the series-parallel st-dag.

A series-parallel st-dag G is naturally associated with a rooted binary tree $\mathcal{T}$ called *binary decomposition tree*. Each leaf of the tree represents an arc in the series-parallel st-dag, each internal node is marked S or P and represents the series or parallel composition of the series-parallel st-dags represented by subtrees rooted at the children of the node. A subgraph of G corresponding to node $\delta \in \mathcal{T}$, denoted by G_δ, is called *pertinent st-dag of* δ. Tree $\mathcal{T}$ is defined as follows (see Figure 1b):

- If G is a single arc, then $\mathcal{T}$ consists a single node.
- If G is created by the parallel composition of series-parallel st-dags G_1 and G_2, let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the decomposition trees of G_1 and G_2, respectively. The root of $\mathcal{T}$ is marked P (the order of subtrees is arbitrary).
- If G is created by the series composition of series-parallel st-dags G_1 and G_2, let $\mathcal{T}_1$ and $\mathcal{T}_2$ be the decomposition trees of G_1 and G_2, respectively. The root of $\mathcal{T}$ is marked S and has left subtree $\mathcal{T}_1$ and right subtree $\mathcal{T}_2$.

We are interested in two kinds of reductions of an st-dag: *parallel reduction* and *series reduction*. A *parallel reduction* at u, v replaces two parallel arcs e and f joining u and v by a single arc $g = (u, v)$ (Figure 2a). A *series reduction* at v is possible if $in(v) = out(v) = 1$ then $e = (u, v)$ and $f = (v, w)$ are replaced by $g = (u, w)$ (Figure 2b).

Valdes et al. [16] give an alternative characterization of series-parallel st-dags based on the reductions.

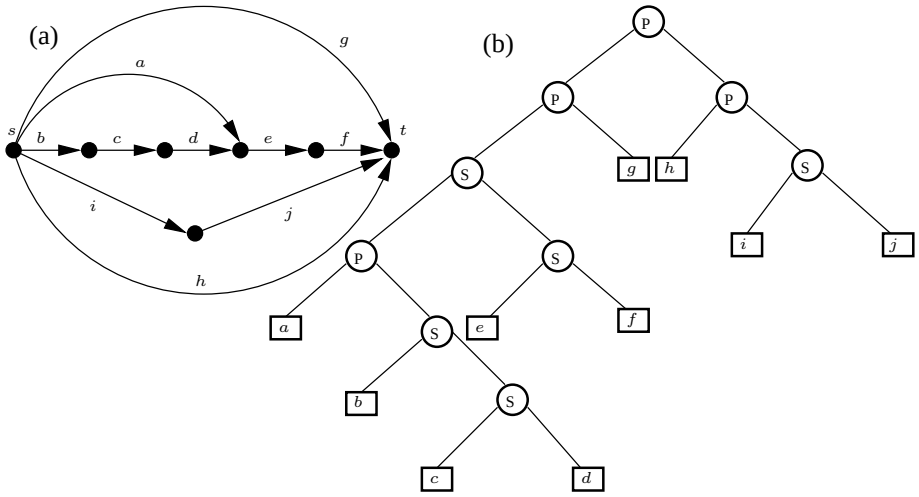

Figure 1: (a) A series-parallel st-dag. (b) Its binary decomposition tree.

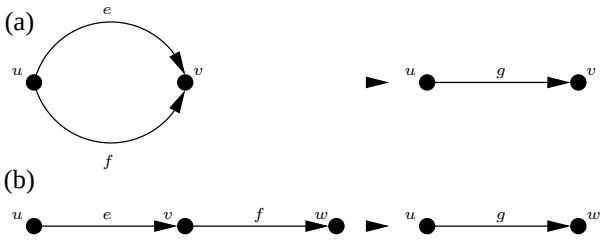

Figure 2: (a) Parallel reduction. (b) Series reduction.

Lemma 1. *Let $[G]$ denote the graph that results when a sequence of all possible series and parallel reductions have been applied to G. An st-dag G is series-parallel, if and only if $[G]$ consists of a single arc.*

An st-dag G is said to be *irreducible* if $[G] = G$. The above characterization suggests an efficient algorithm to recognize series-parallel st-dags. Valdes et al. [16] have provided $\mathcal{O}(m)$ algorithm. In the case when the reduction process succeeds, as a by-product a binary decomposition tree $\mathcal{T}$ of original graph G may be obtained.

3 Problem description

An st-dag $G =< V, A >$ being a project activity-on-arc model is given. V is the set of events (nodes) and A is the set of activities (arcs). Activity durations (weights of the arcs) are to be chosen from intervals $T_e = [\underline{t}_e, \overline{t}_e]$, $e \in A$. The intervals express ranges of possible realizations of the duration times.

We introduce some additional notation. Let T denote a configuration of activity durations $t_e \in T_e$, $e \in A$, while t_e^T denotes the duration of activity e in configuration T, and let P_{ij} be the set of all the paths in G from i to j. We denote the length of a path $p \in P_{ij}$ in configuration T by l_p^T, $l_p^T = \sum_{e \in p} t_e^T$. We use $\mathfrak{T}(A)$ to denote the set

of possible configurations of the activity durations, i.e. $\mathfrak{T}(A)$ is the Cartesian product of corresponding intervals T_e, $e \in A$.

Consider the problem of determining the interval F_ω (bounds) of possible values of floats (total floats) of a distinguished activity $\omega = (\iota, \kappa)$, $\omega \in A$ that has been originally stated in [8], [9]. The interval $F_\omega = [\underline{f}_\omega, \overline{f}_\omega]$ is formed by $\underline{f}_\omega = \min f_\omega^T$ and $\overline{f}_\omega = \max f_\omega^T$, where $\min$ and $\max$ are taken over all possible configurations of the activity durations $\mathfrak{T}(A)$. f_ω^T is the float (total floats) of activity ω in configuration T. Float f_ω^T is determined by means of formula

$$f_\omega^T = \max_{p \in P_{st}} l_p^T - \max_{p \in P_{s\iota}} l_p^T - t_\omega^T - \max_{p \in P_{\kappa t}} l_p^T. \tag{1}$$

We see that the float of activity ω in (1) is expressed as a function of the lengths of paths from s to t, i.e. its float is the difference between the length of the longest path from s to t and the length of the longest path from s to t that traverses ω. A float $f_\omega \in F_\omega$ if and only if there is a configuration of times T such that $f_\omega = f_\omega^T$.

The topic of this paper is computing bounds on floats of all the activities in G under the assumption that G is a series-parallel st-dag.

4 Computing bounds on floats of all activities in a series-parallel st-dag

4.1 An $\mathcal{O}(n^2)$ algorithm

In this section we give an $\mathcal{O}(n^2)$ algorithm for computing bounds on floats of all the activities in series-parallel st-dag. Before we do this, we will provide an $\mathcal{O}(n)$ algorithm, which is invoked as a subroutine, to determine the bounds on floats of a single distinguished activity in series-parallel st-dag using reductions - series and parallel (Figure 2a, b).

The idea of our algorithms is based mainly on the fact that series and parallel reductions preserve the bounds on floats of a distinguished activity. We first prove this.

Let $G' =< V', A' >$ denote the st-dag obtained from the st-dag G by applying one a series reduction or one a parallel reduction, while ω and ω' stand for distinguished activities in G and in G', respectively. To obtain G'; the duration intervals of the new arcs are defined; a new distinguished activity ω' is defined; and factors $\underline{\Delta}$ and $\overline{\Delta}$ are defined; all such that $\underline{f}_\omega = \underline{\Delta} + \underline{f}_{\omega'}$, $\overline{f}_\omega = \overline{\Delta} + \overline{f}_{\omega'}$.

A series reduction replaces arcs e and f with g, $A' = A \cup \{g\} \setminus \{e, f\}$, $V' = V \setminus \{v\}$,

$$\omega' = \begin{cases} g & \text{if } e = \omega \text{ or } f = \omega, \\ \omega & \text{otherwise.} \end{cases} \tag{2}$$

Case: $e = \omega$

$$[\underline{t}_g, \overline{t}_g] = \begin{cases} [\overline{t}_e + \overline{t}_f, \overline{t}_e + \overline{t}_f] & \text{when } \underline{f}_\omega \text{ is computed,} \\ [\underline{t}_e + \underline{t}_f, \underline{t}_e + \underline{t}_f] & \text{when } \overline{f}_\omega \text{ is computed,} \end{cases} \tag{3}$$

$$\underline{\Delta} = 0, \ \overline{\Delta} = 0. \tag{4}$$

Case: $f = \omega$. Similar to $e = \omega$.
Case: $e \neq \omega$ and $f \neq \omega$.

$$[\underline{t}_g, \overline{t}_g] = [\underline{t}_e + \underline{t}_f, \overline{t}_e + \overline{t}_f],\ \underline{\Delta} = 0,\ \overline{\Delta} = 0. \tag{5}$$

A parallel reduction replaces arcs e and f with g, $A^{'} = A \cup \{g\} \setminus \{e, f\}$, $V^{'} = V$,

$$\omega^{'} = \begin{cases} g & \text{if } e = \omega \text{ or } f = \omega, \\ \omega & \text{otherwise.} \end{cases} \tag{6}$$

Case: $e = \omega$

$$[\underline{t}_g, \overline{t}_g] = \begin{cases} [\max\{\overline{t}_e, \underline{t}_f\}, \max\{\overline{t}_e, \underline{t}_f\}] & \text{when } \underline{f}_\omega \text{ is computed,} \\ [\max\{\underline{t}_e, \overline{t}_f\}, \max\{\underline{t}_e, \overline{t}_f\}] & \text{when } \overline{f}_\omega \text{ is computed,} \end{cases} \tag{7}$$

$$\underline{\Delta} = \max\{0, \underline{t}_f - \overline{t}_e\},\ \overline{\Delta} = \max\{0, \overline{t}_f - \underline{t}_e\}. \tag{8}$$

Case: $f = \omega$. Similar to $e = \omega$. It suffices to replace e with f in (7) and (8).
Case: $e \neq \omega$ and $f \neq \omega$

$$[\underline{t}_g, \overline{t}_g] = [\max\{\underline{t}_e, \underline{t}_f\}, \max\{\overline{t}_e, \overline{t}_f\}],\ \underline{\Delta} = 0,\ \overline{\Delta} = 0. \tag{9}$$

The next proposition shows that series and parallel reductions preserve the bounds on floats of a distinguished activity. It is the key to constructing a linear time algorithm for computing bounds on floats.

Proposition 1. *Suppose the st-dag G (not necessarily series-parallel) admits to perform a series reduction or a parallel reduction. Let $G^{'}$ be the st-dag obtained from G by applying a series reduction or a parallel reduction, let ω and $\omega^{'}$ be distinguished activities in G and in $G^{'}$, respectively, and $\underline{\Delta}$ and $\underline{\Delta}$ factors, all as in (2)–(9). Then $\underline{f}_\omega = \underline{\Delta} + \underline{f}_{\omega'}$ and $\overline{f}_\omega = \overline{\Delta} + \overline{f}_{\omega'}$.*

Proof. We need to show that

$$\min_{T \in \mathfrak{T}(A)} f_\omega^T = \underline{\Delta} + \min_{T \in \mathfrak{T}(A')} f_{\omega'}^T \ \ (\text{resp. } \max_{T \in \mathfrak{T}(A)} f_\omega^T = \overline{\Delta} + \max_{T \in \mathfrak{T}(A')} f_{\omega'}^T).$$

Suppose $G^{'}$ is obtained from G by a series reduction. Therefore there are in G activities $e = (u, v)$ and $f = (v, w)$ such that $in(v) = 1$ and $out(v) = 1$. Then activities e, f are replaced by $g = (u, w)$, $g \in A^{'}$ with time interval defined according to (3) or (5).

Consider the case when $\omega = e$ (case $\omega = f$ is similar). So $\omega^{'} = g$ in $G^{'}$ by (2). Let $T^o \in \mathfrak{T}(A)$ be the configuration that minimizes (resp. maximizes) f_ω^T over $\mathfrak{T}(A)$. Thus $\underline{f}_\omega = f_\omega^{T^o}$ (resp. $\overline{f}_\omega = f_\omega^{T^o}$). Since $in(v) = 1$ and $out(v) = 1$, the configuration $T^* \in \mathfrak{T}(A)$ obtained from T^o by increasing (resp. decreasing) duration times $t_e^{T^o}$ and $t_f^{T^o}$ to their upper (resp. lower) bounds is also optimal. In order to get a configuration $T^{'} \in \mathfrak{T}(A^{'})$, it is sufficient to replace $t_e^{T^*}$ and $t_f^{T^*}$ with $t_e^{T^*} + t_f^{T^*}$ in configuration T^*. Now the distinguished activity g in $G^{'}$ has duration time $t_g^{T^{'}}$ equal to $t_e^{T^*} + t_f^{T^*}$, $t_g^{T^{'}} \in [\overline{t}_e + \overline{t}_f, \overline{t}_e + \overline{t}_f]$ (resp. $t_g^{T^{'}} \in [\underline{t}_e + \underline{t}_f, \underline{t}_e + \underline{t}_f]$). Float $f_g^{T^{'}} = f_e^{T^*}$ because the length of the longest path crossing e and f in configuration T^*, in graph G, equals the length of the longest path traversing g in $T^{'}$, in $G^{'}$, and the length of the longest

path in T^* is equal to the length of the longest path in $T^{'}$. Our task is now to show that $T^{'}$ minimizes (resp. maximizes) f_g^T over $\mathfrak{T}(A^{'})$. Suppose on the contrary that there is $T^{''} \in \mathfrak{T}(A^{'})$, $T^{''} \neq T^{'}$, such that $f_g^{T^{''}} < f_g^{T^{'}}$ (resp. $f_g^{T^{''}} > f_g^{T^{'}}$). By (3) and $T^{''} \in \mathfrak{T}(A^{'})$, $t_g^{T^{''}} \in [\overline{t}_e + \overline{t}_f, \overline{t}_e + \overline{t}_f]$ (resp. $t_g^{T^{''}} \in [\underline{t}_e + \underline{t}_f, \underline{t}_e + \underline{t}_f]$). To obtain a configuration $T^{**} \in \mathfrak{T}(A)$, we now replace $t_g^{T^{''}}$ with $\overline{t}_e$ and $\overline{t}_f$ (resp. $\underline{t}_e$ and $\underline{t}_f$). Clearly, $f_g^{T^{''}} = f_e^{T^{**}}$. Since $f_g^{T^{''}} < f_g^{T^{'}}$ (resp. $f_g^{T^{''}} > f_g^{T^{'}}$) and $f_g^{T^{'}} = f_e^{T^*}$, we have $f_e^{T^{**}} < f_e^{T^*}$ (resp. $f_e^{T^{**}} > f_e^{T^*}$), which contradicts the optimality of T^*. Therefore, $\underline{f}_e = \underline{f}_g$ (resp.$\overline{f}_e = \overline{f}_g$) and $\underline{\Delta} = 0$ (resp. $\overline{\Delta} = 0$).

Consider the case when $\omega \neq e$ and $\omega \neq f$. By (2) $\omega^{'} = \omega$ in $G^{'}$. Let $T^* \in \mathfrak{T}(A)$ be the configuration that minimizes (resp. maximizes) f_ω^T, $T \in \mathfrak{T}(A)$. Let us replace $t_e^{T^*}$ and $t_f^{T^*}$ with $t_e^{T^*} + t_f^{T^*}$ in T^*. In this new configuration $T^{'} \in \mathfrak{T}(A^{'})$ $t_g^{T^{'}} = t_e^{T^*} + t_f^{T^*}$. Float $f_{\omega'}^{T^{'}} = f_\omega^{T^*}$, since the lengths of longest paths in configuration T^* and $T^{'}$ remain unchanged. Similarly to the case when $\omega = e$, we can see that $T^{'}$ minimizes (resp. maximizes) $f_{\omega'}^T$ over $\mathfrak{T}(A^{'})$. Thus, $\underline{f}_\omega = \underline{f}_{\omega'}$ (resp.$\overline{f}_\omega = \overline{f}_{\omega'}$) and $\underline{\Delta} = 0$ (resp. $\overline{\Delta} = 0$).

Now suppose $G^{'}$ is obtained from G by a parallel reduction. Therefore there exist in G parallel activities $e = (u, v)$ and $f = (u, v)$. Then activities e, f are replaced by $g = (u, v)$, $g \in A^{'}$ with time interval defined according to (7) or (9).

Consider the case when $\omega = e$ (the same reasoning applies to case $\omega = f$). Hence $\omega^{'} = g$ in $G^{'}$ by (6). Let $T^o \in \mathfrak{T}(A)$ be the configuration that minimizes (resp. maximizes) f_ω^T over $\mathfrak{T}(A)$, i.e. $\underline{f}_\omega = f_\omega^{T^o}$ (resp. $\overline{f}_\omega = f_\omega^{T^o}$). Since e and f leave the same node u and enter the same node v, it follows that the configuration $T^* \in \mathfrak{T}(A)$ obtained from T^o by increasing (resp. decreasing) $t_e^{T^o}$ and decreasing (resp. increasing) $t_f^{T^o}$ to the upper and lower (resp. the lower and upper) bounds, respectively, is also optimal. Observe that if $t_e^{T^*} < t_f^{T^*}$ then $f_e^{T^*} = t_f^{T^*} - t_e^{T^*} + f_f^{T^*}$. Our next goal is to determine the configuration $T^{'} \in \mathfrak{T}(A^{'})$. Replacing $t_e^{T^*}$ and $t_f^{T^*}$ with $\max\{t_e^{T^*}, t_f^{T^*}\}$ in T^* yields $T^{'}$. Now $t_g^{T^{'}} = \max\{t_e^{T^*}, t_f^{T^*}\}$, $t_g^{T^{'}} \in [\max\{\overline{t}_e, \underline{t}_f\}, \max\{\overline{t}_e, \underline{t}_f\}]$ (resp. $t_g^{T^{'}} \in [\max\{\underline{t}_e, \overline{t}_f\}, \max\{\underline{t}_e, \overline{t}_f\}]$). If $t_e^{T^*} < t_f^{T^*}$ then $f_f^{T^*} = f_g^{T^{'}}$. Consequently $f_e^{T^*} = t_f^{T^*} - t_e^{T^*} + f_g^{T^{'}}$. Otherwise $f_e^{T^*} = f_g^{T^{'}}$. This follows from the fact that the length of longest paths have been unchanged in configurations T^* and $T^{'}$. We next claim that $T^{'}$ minimizes (resp. maximizes) f_g^T over $\mathfrak{T}(A^{'})$. Suppose, contrary to our claim, that there is $T^{''} \in \mathfrak{T}(A^{'})$, $T^{''} \neq T^{'}$, such that $f_g^{T^{''}} < f_g^{T^{'}}$ (resp. $f_g^{T^{''}} > f_g^{T^{'}}$). $t_g^{T^{''}} \in [\max\{\overline{t}_e, \underline{t}_f\}, \max\{\overline{t}_e, \underline{t}_f\}]$ (resp. $t_g^{T^{''}} \in [\max\{\underline{t}_e, \overline{t}_f\}, \max\{\underline{t}_e, \overline{t}_f\}]$) by (7) and $T^{''} \in \mathfrak{T}(A^{'})$. Replacing $t_g^{T^{''}}$ with $\overline{t}_e$ and $\underline{t}_f$ (resp. $\underline{t}_e$ and $\overline{t}_f$) gives the configuration $T^{**} \in \mathfrak{T}(A)$. Obviously, $f_g^{T^{''}} = f_f^{T^{**}}$ and so $f_g^{T^{''}} = f_e^{T^{**}} - (\underline{t}_f - \overline{t}_e)$ (resp. $f_g^{T^{''}} = f_e^{T^{**}} - (\overline{t}_f - \underline{t}_e)$) if $\overline{t}_e < \underline{t}_f$ (resp. $\underline{t}_e < \overline{t}_f$). Otherwise $f_g^{T^{''}} = f_e^{T^{**}}$. Consequently, $f_e^{T^{**}} < f_e^{T^*}$ (resp. $f_e^{T^{**}} > f_e^{T^*}$), which contradicts the optimality of T^*. Thus $\underline{f}_e = \underline{\Delta} + \underline{f}_g$ (resp.$\overline{f}_e = \overline{\Delta} + \overline{f}_g$) and $\underline{\Delta} = \max\{0, \underline{t}_f - \overline{t}_e\}$ (resp. $\overline{\Delta} = \max\{0, \overline{t}_f - \underline{t}_e\}$).

We turn to the case when $\omega \neq e$ and $\omega \neq f$. By (6) $\omega^{'} = \omega$ in $G^{'}$. Let $T^* \in \mathfrak{T}(A)$

be the configuration that minimizes (resp. maximizes) f_{ω}^{T}, $T \in \mathfrak{T}(A)$. We replace $t_e^{T^*}$ and $t_f^{T^*}$ with $\max\{t_e^{T^*}, t_f^{T^*}\}$ in T^* to get $T^{'} \in \mathfrak{T}(A^{'})$. Hence $t_g^{T^{'}} = \max\{t_e^{T^*}, t_f^{T^*}\}$. Float $f_{\omega'}^{T^{'}} = f_{\omega}^{T^*}$, since the lengths of longest paths in configuration T^* and $T^{'}$ remain unchanged. The fact that $T^{'}$ minimizes (resp. maximizes) $f_{\omega'}^{T}$ over $\mathfrak{T}(A^{'})$ follows in the same way as in the case $\omega = e$. Therefore, $\underline{f}_{\omega} = \underline{f}_{\omega'}$ (resp.$\overline{f}_{\omega} = \overline{f}_{\omega'}$) and $\underline{\Delta} = 0$ (resp. $\overline{\Delta} = 0$). □

Now we will build up a linear time algorithm for determining bounds on floats, $\underline{f}_{\omega}$ and $\overline{f}_{\omega}$, of a distinguished activity ω using series and parallel reductions. This algorithm can be outlined as follows: step 1: make a series or parallel reduction; step 2: repeat step 1 until G is reduced to a single arc. If G is originally series-parallel st-dag then Lemma 1 guarantees that the above algorithm reduces G to a single arc. Bounds on floats $\underline{f}_{\omega}$ and $\overline{f}_{\omega}$ are determined by initializing $\underline{S} \leftarrow 0$ and $\overline{S} \leftarrow 0$, applying a series or parallel reduction to some arcs e and f, defining an arc interval of a new arc g, letting $\underline{S} \leftarrow \underline{S} + \underline{\Delta}$ and $\overline{S} \leftarrow \overline{S} + \overline{\Delta}$ if a parallel reduction is made and $e = \omega$ or $f = \omega$. At the end, i.e. if the reduced graph consists of a single arc, the bounds on floats of distinguished activity ω in the original graph are given by $\underline{f}_{\omega} = \underline{S}$ and $\overline{f}_{\omega} = \overline{S}$. Proposition 1 guarantees that the bounds are properly computed.

The following algorithm is a linear time implementation of the one outlined above (comments are enclosed after $\triangleright$).

Require: A st-dag G with node set V, $|V| \geq 2$, activity set A, $|A| \geq 2$, time intervals T_e for activity $e \in A$, a distinguished activity $\omega \in A$.
Ensure: Bounds $\underline{f}_{\omega}$, $\overline{f}_{\omega}$ if G is series-parallel st-dag or message that G is not series-parallel.

```
S ← 0; S̄ ← 0;
Construct list, L ← {v | v ∈ V, v ≠ s, v ≠ t} marking all such v "onlist";
while T ≠ ∅ and |A| > 1 do
    Remove v from L and mark v "offlist";
    Examine arc (u, v) entering v, distinct ← False;
    while in(v) > 1 and not distinct do
        Examine next arc (x, v) entering v;
        if u = x then ▷ A parallel reduction on arcs e = (u, v) and f = (x, v)
            if (u, v) is "distinguished" or (x, v) is "distinguished" then
                Mark (u, v) "distinguished", and let S ← S + Δ and S̄ ← S̄ + Δ̄;
            end if
            Apply a parallel reduction to (u, v) and (x, v) to obtain (u, v); ▷ g = (u, v)
        else
            distinct ← True;
        end if
    end while
    Examine arc (v, w) leaving v, distinct ← False;
    while out(v) > 1 and not distinct do
        Examine next arc (v, y) leaving v;
        if w = y then ▷ A parallel reduction on arcs e = (v, w) and f = (v, y)
            if (v, w) is "distinguished" or (v, y) is "distinguished" then
                Mark (v, w) "distinguished", and let S ← S + Δ and S̄ ← S̄ + Δ̄;
            end if
            Apply a parallel reduction to (v, w) and (v, y) to obtain (v, w); ▷ g = (v, w)
        else
```

```
        distinct ← True;
      end if
    end while
    if in(v) = 1 and out(v) = 1 then ▷ A series reduction
      Apply a series reduction to delete v and replace (u, v) and (v, w) by a new arc (u, w);
      ▷ g = (u, w)
      if (u, v) is "distinguished" or (v, w) is "distinguished" then
        Mark (u, w) "distinguished";
      end if
      if u ≠ s and u is "offlist" then
        Add u to L and mark u "onlist";
      end if
      if w ≠ t and w is "offlist" then
        Add w to L and mark w "onlist";
      end if
    end if
  end while
  if |A| = 1 then
    print("f_ω =", S, "f̄_ω =", S̄);
  else
    print("G is not series-parallel");
  end if
```

The above algorithm is based on the fundamental work of Valdes et al. [16] where the problem is the recognition of series-parallel st-dags. Therefore its correctness and linear time complexity follow directly from Lemma 1, Proposition 1 and the correctness of an algorithm for recognizing series-parallel st-dag proposed in [16]. In order to make our algorithm running in linear time, we use for each node a doubly linked list of the arcs entering it and a list of the arcs leaving it with corresponding arc intervals. Such representation allows to perform each arc examination, parallel reduction and series reduction (together with defining arc interval) in $\mathcal{O}(1)$ time. Arguments presented in [16] to show that an algorithm for recognizing series-parallel st-dag runs in $\mathcal{O}(m)$ time apply almost verbatim to prove that our algorithm requires $\mathcal{O}(m)$ time.

It is interesting to notice that the presented algorithm can be used, as a subroutine, for a simplification of the problem of computing the bounds on floats of a distinguished activity in a non-series- parallel st-dag, which is $\mathcal{NP}$-hard. Namely, first we call the algorithm to make all possible series and parallel reductions. In this way, the complexity of computing bounds can be often reduced to some degree, sometime we may achieve a considerable simplification. By Proposition 1, the original problem is now equivalent to the problem of computing the bounds on floats of a distinguished activity in a reduced graph.

We now turn to the problem of computing the bounds on floats of all activities in series-parallel st-dag G (the topic of this paper). In order to deal with it we call the presented algorithm, as a subroutine, for each activity in G. The resulting running time is $\mathcal{O}(m^2)$. Note that for series-parallel st-dags $m = \mathcal{O}(n)$ and consequently the running time is $\mathcal{O}(n^2)$. If instead of our algorithm we call the one proposed in [9] then the running time for this version is $\mathcal{O}(n^2)$. So, both approaches require the same time. In Section 4.2, we give an $\mathcal{O}(n \log n)$ algorithm for the considered problem.

4.2 An $\mathcal{O}(n \log n)$ algorithm

In this section we will assume that an interval weighted st-dag $G =< V, A >$ being a project activity-on-arc model is series-parallel (see Figure 3a). We still deal with the problem of determining the bounds on floats of all activities in G. A solution to this problem has been provided by Fargier et al. [9] (see also Dubois et al. [8]) as we mention in Section 4.1. It has been based on the following theorem that allows to determine the configuration that minimizes (resp. maximizes) quantity (1), i.e. the float of given activity, in a series-parallel st-dag.

Theorem 1. *Let G be the series-parallel st-dag, let $\omega = (\iota, \kappa)$, $\omega \in A$, be a distinguished activity, and $T^*(\omega)$ and $T_*(\omega)$, $T^*(\omega), T_*(\omega) \in \mathfrak{T}(A)$, are the following configurations:*

$$t_e^{T^*(\omega)} = \begin{cases} \overline{t}_e & \text{if} \quad e \in A_{s\iota} \cup \{\omega\} \cup A_{\kappa t}, \\ \underline{t}_e & \text{otherwise.} \end{cases}$$

$$t_e^{T_*(\omega)} = \begin{cases} \underline{t}_e & \text{if} \quad e \in A_{s\iota} \cup \{\omega\} \cup A_{\kappa t}, \\ \overline{t}_e & \text{otherwise,} \end{cases}$$

where $A_{s\iota}$ (resp. $A_{\kappa t}$) stands for the set of all the activities that belong to the paths from set $P_{s\iota}$ (resp. $P_{\kappa t}$). Then $T^*(\omega)$ minimizes and $T_*(\omega)$ maximizes the float of ω.

Theorem 1 shows that computing bounds on floats of a single distinguished activity ω boils down to two invocations of the CPM, which requires $\mathcal{O}(n)$ time (for series-parallel st-dags $m = \mathcal{O}(n)$) and thus an algorithm provided by Fargier et al. [9] takes $\mathcal{O}(n)$ time. Consequently, computing bounds on floats of all activities in G requires $\mathcal{O}(n^2)$ time.

Our solution to the considered problem is also based on Theorem 1. It consists in determining implicitly two optimal configurations and computing the floats in these configurations for each activity in G. By introducing a data structure inspired by dynamic expression trees of Cohen and Tamassia [2] and dynamic trees of Sleator and Tarjan [15], we can reduce the time for computing bounds on floats to $\mathcal{O}(\log n)$ for each activity in G and thus computing bounds on floats of all the activities in G requires $\mathcal{O}(n \log n)$ time. In our presentation, we follow the notation of [2].

A series-parallel st-dag G may be represented by binary decomposition tree $\mathcal{T}$, whose each node δ corresponds to the pertinent st-dag G_δ of δ. This tree can be constructed in $\mathcal{O}(n)$ as we mention at the end of Section 2. Two lengths are associated with each node δ of $\mathcal{T}$: the length of the longest path from the source $s(G_\delta)$ to the sink $t(G_\delta)$ of the pertinent st-dag G_δ of δ, denoted by $lolo(\delta)$, in which duration times are at their lower bounds and the length of a longest $s(G_\delta)t(G_\delta)$-path of G_δ, denoted by $uplo(\delta)$, in which duration times are at their upper bounds (values in square brackets in Figure 3b). If node δ is a leaf, which corresponds to arc e in G, then $lolo(\delta) = \underline{t}_e$ and $uplo(\delta) = \overline{t}_e$. Lengths $lolo(\delta)$ and $uplo(\delta)$ are determined by the following formulae:

$$lolo(\delta) = \begin{cases} \max\limits_{\gamma \in children(\delta)} lolo(\gamma) & \text{if } \delta \text{ is marked P,} \\ \\ \sum\limits_{\gamma \in children(\delta)} lolo(\gamma) & \text{if } \delta \text{ is marked S,} \end{cases} \tag{10}$$

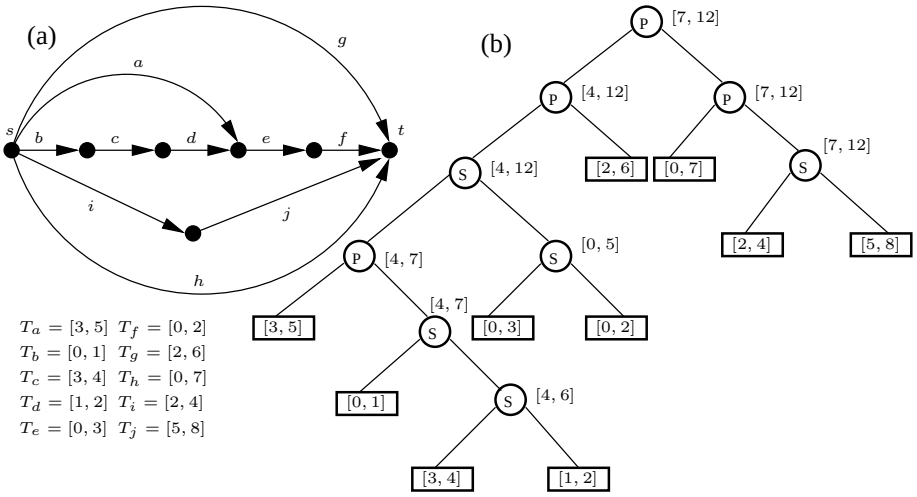

Figure 3: (a) An interval weighted series-parallel st-dag G. (b) Its binary decomposition tree $\mathcal{T}$. Shown at each node δ of $\mathcal{T}$ is the interval being lengths of longest $s(G_\delta)t(G_\delta)$-paths of corresponding the pertinent st-dag G_δ of δ, in which all duration times are at their lower bounds and all duration times are at their upper bounds.

$$uplo(\delta) = \begin{cases} \max\limits_{\gamma \in children(\delta)} uplo(\gamma) & \text{if } \delta \text{ is marked P,} \\ \\ \sum\limits_{\gamma \in children(\delta)} uplo(\gamma) & \text{if } \delta \text{ is marked S,} \end{cases} \tag{11}$$

where $children(\delta)$ denotes the set of children of node δ in $\mathcal{T}$.

Now we characterize the dependence of the length of the longest st-path in st-dag G, in some time configurations, on the duration time of activity $e \in A$ or more generally on the length of the longest st-path in its subgraph.

Consider two pertinent st-dags $G_\epsilon =< V_\epsilon, A_\epsilon >$ and $G_\delta =< V_\delta, A_\delta >$ of ϵ and δ such that $V_\delta \subseteq V_\epsilon \subseteq V$, $A_\delta \subseteq A_\epsilon \subseteq A$ and the corresponding decomposition tree $\mathcal{T}$ of G rooted at ρ. From this, it may be concluded that node δ is one of the descendants of node ϵ in $\mathcal{T}$ (there is a path from δ to ϵ). The dependence of length $lolo(\epsilon)$ on length $lolo(\delta)$ is characterized by pair $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$ and so

$$lolo(\epsilon) = \max\{lolo(\delta) + \underline{\mathfrak{a}}, \underline{\mathfrak{b}}\}.$$

We assign the $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$ to the path from δ to ϵ. Note that this pair is different for each path in $\mathcal{T}$. If node δ is a leaf of $\mathcal{T}$ associated with activity $\omega \in A$ then the length of the longest st-path of G in which duration times are at their lower bounds can be determined by the following formula:

$$lolo(\rho) = \max\{lolo(\delta) + \underline{\mathfrak{a}}, \underline{\mathfrak{b}}\}. \tag{12}$$

Let us denote this configuration of times by $\underline{T}$, $\underline{T} \in \mathfrak{T}(A)$. Note also that now we consider the path from δ to ρ and $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$ is assigned to it. From (12) we conclude that activity ω is *critical* (it is on the longest st-path of G) in $\underline{T}$ (or equivalently increasing

the duration time of ω effects the length of the longest st-path of G) if and only if $t_{\omega}^{\underline{T}} \geq \underline{\mathfrak{b}} - \underline{\mathfrak{a}}$, where $t_{\omega}^{\underline{T}} = \underline{t}_{\omega}$. Hence, we have the float of ω in $\underline{T}$

$$f_{\omega}^{\underline{T}} = \max\{0, \underline{\mathfrak{b}} - \underline{\mathfrak{a}} - \underline{t}_{\omega}\}. \tag{13}$$

Likewise, the dependence of length $uplo(\epsilon)$ on length $uplo(\delta)$ is characterized by pair $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$ and we have

$$uplo(\epsilon) = \max\{uplo(\delta) + \overline{\mathfrak{a}}, \overline{\mathfrak{b}}\}.$$

If node δ is a leaf of $\mathcal{T}$ associated with activity $\omega \in A$ then the float of ω in configuration $\overline{T} \in \mathfrak{T}(A)$ in which duration times are at their upper bounds can be determined similarly to (13).

Let us focus on crucial dependences for a solution to the problem considered in this section. The dependence of length of the longest $s(G_{\epsilon})t(G_{\epsilon})$-path of st-dag G_{ϵ}, denoted by $lo_*(\epsilon)$, in which the duration times of activities belonging to the set $A_{s(G_{\epsilon})s(G_{\delta})} \cup A_{t(G_{\delta})t(G_{\epsilon})}$ are at their lower bounds and the duration times of activities belonging to the set $A_{\epsilon} \setminus (A_{s(G_{\epsilon})s(G_{\delta})} \cup A_{\delta} \cup A_{t(G_{\delta})t(G_{\epsilon})})$ are at their upper bounds on the length of the longest $s(G_{\delta})t(G_{\delta})$-path, denoted by $lo(\delta)$, of st-dag G_{δ} in which duration times may be arbitrary chosen from time intervals is characterized by pair $(\mathfrak{a}_*, \mathfrak{b}_*)$ and we obtain

$$lo_*(\epsilon) = \max\{lo(\delta) + \mathfrak{a}_*, \mathfrak{b}_*\}.$$

When node δ is a leaf of $\mathcal{T}$ associated with activity ω and additionally $lo(\delta) = \underline{t}_{\omega}$ (now we consider the path from leaf δ to the root ρ of $\mathcal{T}$ with pair $(\mathfrak{a}_*, \mathfrak{b}_*)$ associated to it). Then $lo_*(\rho)$ is the length of the longest st-path of G in configuration $T_*(\omega)$ (that from Theorem 1) and can be calculated as follows

$$lo_*(\rho) = \max\{lo(\delta) + \mathfrak{a}_*, \mathfrak{b}_*\}. \tag{14}$$

From (14) and fact that $lo(\delta) = \underline{t}_{\omega}$, we deduce that ω is critical in $T_*(\omega)$ if and only if $\underline{t}_{\omega} \geq \mathfrak{b}_* - \mathfrak{a}_*$. The above and Theorem 1 give the upper bound on the float of ω

$$\overline{f}_{\omega} = f_{\omega}^{T_*(\omega)} = \max\{0, \mathfrak{b}_* - \mathfrak{a}_* - \underline{t}_{\omega}\}. \tag{15}$$

Similarly, the dependence of the length of the longest $s(G_{\epsilon})t(G_{\epsilon})$-path of st-dag G_{ϵ}, denoted by $lo^*(\epsilon)$, in which the duration times of activities belonging to the set $A_{s(G_{\epsilon})s(G_{\delta})} \cup A_{t(G_{\delta})t(G_{\epsilon})}$ are at their upper bounds and the duration times of activities belonging to the set $A_{\epsilon} \setminus (A_{s(G_{\epsilon})s(G_{\delta})} \cup A_{\delta} \cup A_{t(G_{\delta})t(G_{\epsilon})})$ are at their lower bounds from the length $lo(\delta)$ is characterized by pair $(\mathfrak{a}^*, \mathfrak{b}^*)$ and we get

$$lo^*(\epsilon) = \max\{lo(\delta) + \mathfrak{a}^*, \mathfrak{b}^*\}.$$

Consider the case when node δ is a leaf of $\mathcal{T}$ associated with activity ω and additionally $lo(\delta) = \overline{t}_{\omega}$ (now we focus on the path from leaf δ to the root ρ of $\mathcal{T}$). Then $lo^*(\rho)$ is the length of the longest st-path of G in configuration $T^*(\omega)$ (that from Theorem 1) can be determined as follows

$$lo^*(\rho) = \max\{lo(\delta) + \mathfrak{a}^*, \mathfrak{b}^*\}. \tag{16}$$

Pair $(\mathfrak{a}^*, \mathfrak{b}^*)$ is assigned to the path from δ to ρ. Similar arguments to that for the upper bounds of ω apply here and thus we have the lower bound on the float of ω

$$\underline{f}_\omega = f_\omega^{\mathcal{T}^*(\omega)} = \max\{0, \mathfrak{b}^* - \mathfrak{a}^* - \overline{t}_\omega\}. \tag{17}$$

Pairs $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ are assigned to the path from δ to ϵ in $\mathcal{T}$ and they are different for each path in $\mathcal{T}$. We will show in the description of $join$ operation how to maintain and determine these values.

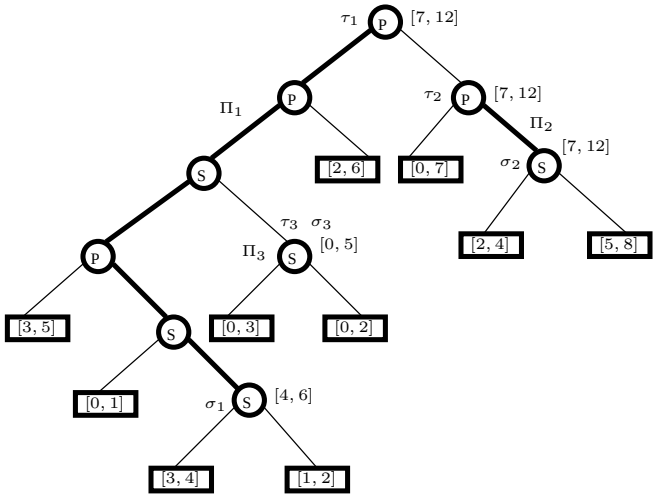

Figure 4: Solid paths for the decomposition tree of Figure 3b. Solid edges are shown in bold. Π_3 and each leaf are one-node solid paths.

We store a binary decomposition tree $\mathcal{T}$ that describes the structure of a series-parallel st-dag G in a data structure for dynamic trees [15]. Our approach is similar in spirit to that for dynamic expression trees [2] . We partition tree $\mathcal{T}$ into a collection of node-disjoint paths. (We use partitioning by size as in [15]) Namely, we partition the edges of $\mathcal{T}$ into two kinds: *solid* and *dashed*. It is assumed that edges of $\mathcal{T}$ are directed from the child to the parent. An edge (δ, ϵ) of $\mathcal{T}$ is called to be solid if the subtree rooted at δ has more then half of the nodes of the subtree rooted at its parent ϵ. Otherwise, (δ, ϵ) is called to be dashed. Thus at most one solid edge enters any node. The solid edges define a collection of *solid paths* that partition the nodes. A node with no incident solid edge is a one-node solid path (see Figure 4). The head σ of a solid path Π is its bottommost node; the tail τ is its topmost node. Each solid path Π is represented by a balanced binary tree $\mathcal{B}_\Pi$ (see Figure 5). We call it *path tree*. Therefore, $\mathcal{T}$ is kept as a collection of path trees. Each solid path Π always satisfies the following invariant (see Figure 4).

PATH INVARIANT. *The tail τ and head σ of each solid path Π store the actual values $lolo(\tau)$, $uplo(\tau)$ and $lolo(\sigma)$, $uplo(\sigma)$.*

The above values are easy to find. If σ is a leaf of $\mathcal{T}$ associated with $e \in A$ then $lolo(\sigma) = \underline{t}_e$ and $uplo(\sigma) = \overline{t}_e$. Otherwise, $lolo(\sigma)$ and $uplo(\sigma)$ are calculated from values $lolo(\tau^{'})$, $uplo(\tau^{'})$ and $lolo(\tau^{''})$, $uplo(\tau^{''})$ according to equations (10) and (11), where $\tau^{''}, \tau^{''} \in children(\sigma)$. These values are known, since $\tau^{''}$ and $\tau^{''}$ are tails of

their solid paths, which satisfy path invariant. $lolo(\tau)$ and $uplo(\tau)$ are determined from $lolo(\sigma)$ and $uplo(\sigma)$ by using the path trees.

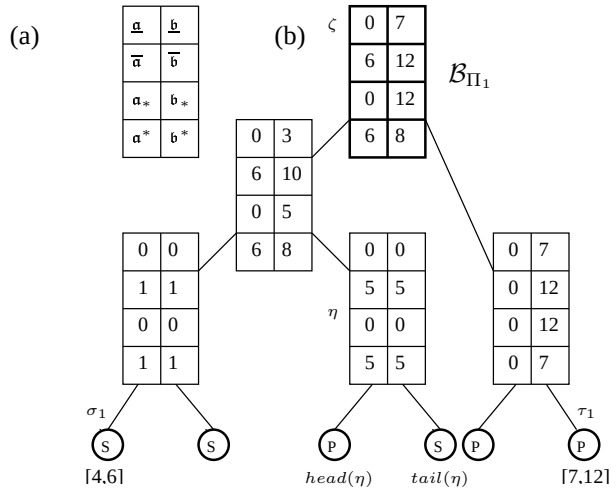

Figure 5: (a) Format of internal nodes. (b) The path tree $\mathcal{B}_{\Pi_1}$ for the solid path Π_1 of Figure 4

Let us consider a solid path Π. The path is represented by path tree $\mathcal{B}_\Pi$ whose leaves of path tree $\mathcal{B}_\Pi$ in left-to-right order correspond to the nodes on the path from the head σ to the tail τ (see Figure 5). Every internal node η in $\mathcal{B}_\Pi$ corresponds to the subpath of Π between its external descendants $head(\eta)$ and $tail(\eta)$. It contains four pairs: $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ such that

$$
\begin{aligned}
lolo(tail(\eta)) &= \max\{lolo(head(\eta)) + \underline{\mathfrak{a}}, \underline{\mathfrak{b}}\},\\
uplo(tail(\eta)) &= \max\{uplo(head(\eta)) + \overline{\mathfrak{a}}, \overline{\mathfrak{b}}\},\\
lo_*(tail(\eta)) &= \max\{lo(head(\eta)) + \mathfrak{a}_*, \mathfrak{b}_*\},\\
lo^*(tail(\eta)) &= \max\{lo(head(\eta)) + \mathfrak{a}^*, \mathfrak{b}^*\}.
\end{aligned}
$$

To clarify, we show how to find $lolo(\tau_1)$ and $uplo(\tau_1)$ of the tail τ_1 of path Π_1 shown in Figure 4. Tree $\mathcal{B}_{\Pi_1}$ shown in Figure 5, rooted at ζ is its path tree. Regarding the root of a path tree is as identifying the solid path and so $\tau_1 = tail(\zeta)$ and $\sigma_1 = head(\zeta)$. Since $\underline{\mathfrak{a}} = 0$, $\underline{\mathfrak{b}} = 7$, $\overline{\mathfrak{a}} = 6$, $\overline{\mathfrak{b}} = 12$, $lolo(\sigma_1) = 4$ and $uplo(\sigma_1) = 6$, we immediately get $lolo(\tau_1) = \max\{0 + 4, 7\} = 7$ and $uplo(\tau_1) = \max\{6 + 6, 12\} = 12$. Note that the obtained values are the lengths of the longest st-paths of G, shown in Figure 3a, in configurations $\underline{T}$ and $\overline{T}$.

Our next goal is to show how to implement an operation that calculates the bounds on floats of a given activity in $\mathcal{O}(\log n)$.

> $Floats(\textbf{\textit{node}}\ \gamma)$: This operation assumes that γ is a leaf of tree $\mathcal{T}$ associated with activity $\omega \in A$. $Floats(\gamma)$ returns a list $[\underline{f}_\omega, \overline{f}_\omega]$, where $\underline{f}_\omega$ and $\overline{f}_\omega$ are the bounds on floats of activity ω.

Using three following operations derived from dynamic expression trees [2] which were derived from dynamic trees [15], we will implement $Floats(\gamma)$:

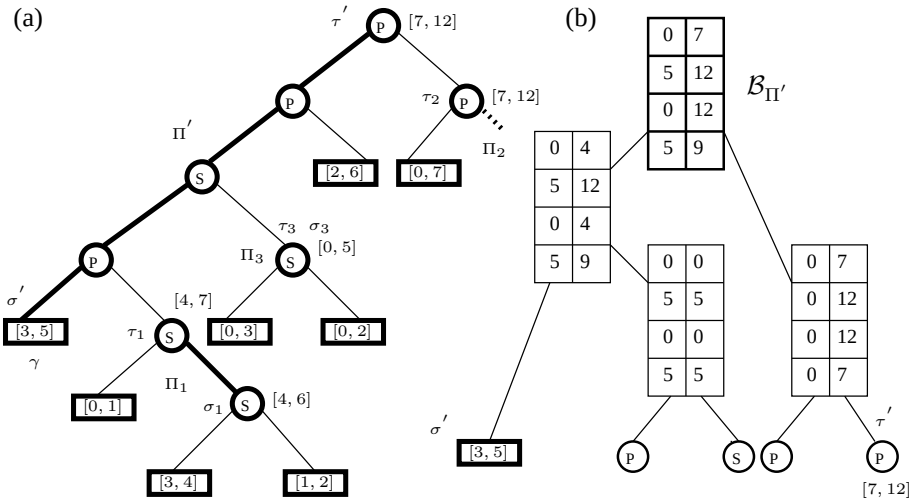

Figure 6: Operation $expose(\gamma)$. Leaf γ is associated with activity a in G. (a) Solid path $\Pi^{'}$ created by $expose(\gamma)$. (b) The path tree $\mathcal{B}_{\Pi'}$ for $\Pi^{'}$.

$splice($***path*** $\Pi)$: Extent the solid path Π by converting the dashed edge leaving the tail τ of Π to solid and converting the solid edge entering the parent ϵ of tail τ (if any) to dashed. This operation assumes that τ is not the root ρ of tree $\mathcal{T}$. Let $\Pi^{'}$ be the solid path containing ϵ. Operation $splice(\Pi)$ is realized as follows. First, we convert dashed edge $(sib(\tau), \epsilon)$ to solid, where $sib(\tau)$ denotes the sibling of τ in tree $\mathcal{T}$. Next we obtain the solid path $\Pi^{''}$ starting at ϵ by splitting $\mathcal{B}_{\Pi'}$. To restore the path invariant, we perform $lolo(sib(\tau))$ and $uplo(sib(\tau))$ and then $lolo(\epsilon)$ and $uplo(\epsilon)$. Finally, we concatenate $\mathcal{B}_{\Pi}$ and $\mathcal{B}_{\Pi''}$. Operation $splice(\Pi)$ returns the extended path.

$expose($***node*** $\delta)$: Create a solid path from δ to the root ρ of $\mathcal{T}$ by converting all dashed edges to solid along the path from δ to ρ and converting solid edges incident to this path to dashed. (See an example of $expose$ in Figure 6 and compare with the tree in Figure 4). Operation $expose(\delta)$ consists of a sequence of $splice$. It returns the resulting solid path. This operation is always followed by $conceal$, which repairs the demage caused by $expose$.

$conceal($***node*** $\delta)$: It is the inverse of $expose$. This operation restores the original type, solid or dashed, of the edges entering the nodes on the path from δ to the root ρ of tree $\mathcal{T}$. It consists of a sequence of $splice$.

In order to implement concatenations and splits of balanced binary trees that represent solid paths, we need three operations:

$join($***node*** $\zeta^{'}, \zeta^{''})$: Given the roots $\zeta^{'}$ and $\zeta^{''}$ of two binary trees $\mathcal{B}_{\Pi'}$ and $\mathcal{B}_{\Pi''}$ representing solid paths $\Pi^{'}$ and $\Pi^{''}$, combine trees into a single tree $\mathcal{B}_{\Pi}$, representing a solid path Π, by constructing a new root ζ with left child $\zeta^{'}$ and right child $\zeta^{''}$. The pairs $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ of ζ are calculated

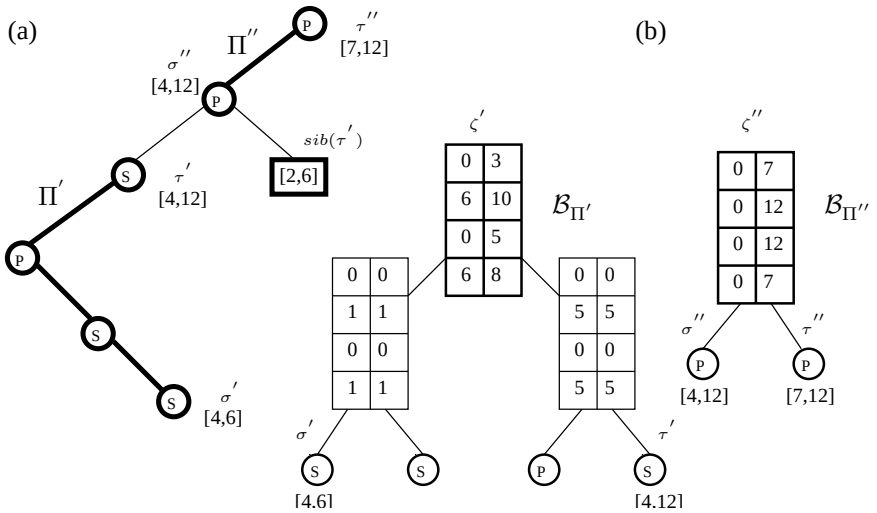

Figure 7: Operation $join(\zeta^{'}, \zeta^{''})$. (a) Solid paths $\Pi^{'}$ and $\Pi^{''}$ linked by dashed edge $(\tau^{'}, \sigma^{''})$. (b) The path trees representing solid paths $\Pi^{'}$ and $\Pi^{''}$. The resulting path tree is given in Figure 5b.

from the pairs $(\underline{\mathfrak{a}}^{'}, \underline{\mathfrak{b}}^{'})$, $(\overline{\mathfrak{a}}^{'}, \overline{\mathfrak{b}}^{'})$, $(\mathfrak{a}_*, \mathfrak{b}^{'}_*)$, $(\mathfrak{a}^{*'}, \mathfrak{b}^{*'})$ of $\zeta^{'}$ and the pairs $(\underline{\mathfrak{a}}^{''}, \underline{\mathfrak{b}}^{''})$, $(\overline{\mathfrak{a}}^{''}, \overline{\mathfrak{b}}^{''})$, $(\mathfrak{a}^{''}_*, \mathfrak{b}^{''}_*)$, $(\mathfrak{a}^{*''}, \mathfrak{b}^{*''})$ of $\zeta^{''}$.

We next derive folmulae for determining $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$. Let $\sigma^{'}$ and $\tau^{'}$ be the head and tail of $\Pi^{'}$ and let $\sigma^{''}$ and $\tau^{''}$ be the head and tail of $\Pi^{''}$ (see Figure 7). After $join$ $\sigma^{'}$ and $\tau^{''}$ become the head and tail of the resulting solid path Π, $\circledast$ is the binary operation such that

$$a \circledast b = \begin{cases} \max\{a, b\} & \text{if } \sigma^{''} \text{ is marked P,} \\ a + b & \text{if } \sigma^{''} \text{ is marked S.} \end{cases}$$

The following dependence holds for $\Pi^{'}$

$$lolo(\tau^{'}) = \max\{lolo(\sigma^{'}) + \underline{\mathfrak{a}}^{'}, \underline{\mathfrak{b}}^{'}\}.$$

Length $lolo(\sigma^{''})$ is calculated from its children

$$lolo(\sigma^{''}) = lolo(\tau^{'}) \circledast lolo(sib(\tau^{'})).$$

Also for $\Pi^{'}$ the following dependence holds

$$lolo(\tau^{''}) = \max\{lolo(\sigma^{''}) + \underline{\mathfrak{a}}^{''}, \underline{\mathfrak{b}}^{''}\}.$$

Combining the above equations yields

$$\begin{aligned} lolo(\tau^{''}) \;\; = \;\; & \max\{\max\{lolo(\sigma^{'}) + \underline{\mathfrak{a}}^{'}, \underline{\mathfrak{b}}^{'}\} \circledast \\ & lolo(sib(\tau^{'})) + \underline{\mathfrak{a}}^{''}, \underline{\mathfrak{b}}^{''}\}. \end{aligned}$$

Making use of the facts that $\max\{a+b, a+c\} = a + \max\{b, c\}$ and $\max\{a, \max\{b, c\}\}$ $= \max\{max\{a, b\}, c\}$ we get the dependence for Π

$$lolo(\tau^{''}) = \max\{lolo(\sigma^{'}) + \underline{\mathfrak{a}}, \underline{\mathfrak{b}}\},$$

where

$$\underline{\mathfrak{a}} = \begin{cases} \underline{\mathfrak{a}}' + \underline{\mathfrak{a}}'' & \text{if } \sigma'' \text{ is marked P,} \\ \underline{\mathfrak{a}}' + \underline{\mathfrak{a}}'' + lolo(sib(\tau')) & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\underline{\mathfrak{b}} = \begin{cases} \max\{\max\{\underline{\mathfrak{b}}', lolo(sib(\tau'))\} + \underline{\mathfrak{a}}'', \underline{\mathfrak{b}}''\} & \text{if } \sigma'' \text{ is marked P,} \\ \max\{\underline{\mathfrak{b}}' + \underline{\mathfrak{a}}'' + lolo(sib(\tau')), \underline{\mathfrak{b}}''\} & \text{if } \sigma'' \text{ is marked S.} \end{cases}$$

Similar consideration holds for $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ and thus we have

$$\overline{\mathfrak{a}} = \begin{cases} \overline{\mathfrak{a}}' + \overline{\mathfrak{a}}'' & \text{if } \sigma'' \text{ is marked P,} \\ \overline{\mathfrak{a}}' + \overline{\mathfrak{a}}'' + uplo(sib(\tau')) & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\underline{\mathfrak{b}} = \begin{cases} \max\{\max\{\overline{\mathfrak{b}}', uplo(sib(\tau'))\} + \overline{\mathfrak{a}}'', \overline{\mathfrak{b}}''\} & \text{if } \sigma'' \text{ is marked P,} \\ \max\{\overline{\mathfrak{b}}' + \overline{\mathfrak{a}}'' + uplo(sib(\tau')), \overline{\mathfrak{b}}''\} & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\mathfrak{a}_* = \begin{cases} \mathfrak{a}_*' + \mathfrak{a}_*'' & \text{if } \sigma'' \text{ is marked P,} \\ \mathfrak{a}_*' + \mathfrak{a}_*'' + lolo(sib(\tau')) & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\mathfrak{b}_* = \begin{cases} \max\{\max\{\mathfrak{b}_*', uplo(sib(\tau'))\} + \mathfrak{a}_*'', \mathfrak{b}_*''\} & \text{if } \sigma'' \text{ is marked P,} \\ \max\{\mathfrak{b}_*' + \mathfrak{a}_*'' + lolo(sib(\tau')), \mathfrak{b}_*''\} & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\mathfrak{a}^* = \begin{cases} \mathfrak{a}^{*'} + \mathfrak{a}^{*''} & \text{if } \sigma'' \text{ is marked P,} \\ \mathfrak{a}^{*'} + \mathfrak{a}^{*''} + uplo(sib(\tau')) & \text{if } \sigma'' \text{ is marked S,} \end{cases}$$

$$\mathfrak{b}^* = \begin{cases} \max\{\max\{\mathfrak{b}^{*'}, lolo(sib(\tau'))\} + \mathfrak{a}^{*''}, \mathfrak{b}^{*''}\} & \text{if } \sigma'' \text{ is marked P,} \\ \max\{\mathfrak{b}^{*'} + \mathfrak{a}^{*''} + uplo(sib(\tau')), \mathfrak{b}^{*''}\} & \text{if } \sigma'' \text{ is marked S.} \end{cases}$$

Note. The values of the pair $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ of a single-node path tree representing one-node solid path are equal to zeros.

An example of $join$ operation on binary trees $\mathcal{B}_{\Pi'}$ and $\mathcal{B}_{\Pi''}$ is given in Figure 7.

$separate($***node*** $\zeta)$: Given the root ζ of a binary tree, divide the tree into two trees with roots ζ' and ζ'', where ζ' is the root of the left subtree and ζ'' is the root of the right subtree. This operation does not modify pairs $(\underline{\mathfrak{a}}, \underline{\mathfrak{b}})$, $(\overline{\mathfrak{a}}, \overline{\mathfrak{b}})$, $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ of ζ' and ζ''.

$rotateleft($***node*** $\eta)$ ($rotateleft($***node*** $\eta)$): Perform a single left (right) rotation at node η. This operation can be implemented by means of $separate$ and $join$.

All the above operations requires $\mathcal{O}(1)$ time.

After these preparations, we are now in a position to implement $Floats$. We implement $Floats($***node*** $\gamma)$ as follows. First, we execute $expose(\gamma)$, which returns solid path Π from δ to the root ρ of tree $\mathcal{T}$. Next, we compute the bounds on float of activity ω associated with γ according to (15) and (17), i.e. $\underline{f}_\omega = \max\{0, \mathfrak{b}^* - \mathfrak{a}^* - \overline{t}_\omega\}$ and $\overline{f}_\omega = \max\{0, \mathfrak{b}_* - \mathfrak{a}_* - \underline{t}_\omega\}$, where $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ correspond to the root ζ of path tree $\mathcal{B}_\Pi$. We complete $float$ by performing $conceal(\gamma)$ to restore the original solid and dashed edges. $Floats$ returns list $[\underline{f}_\omega, \overline{f}_\omega]$. For example, consider the

state after $expose(\gamma)$ given in Figure 6. Operation $expose(\gamma)$ has created solid path Π'. Since the actual values of $(\mathfrak{a}_*, \mathfrak{b}_*)$ and $(\mathfrak{a}^*, \mathfrak{b}^*)$ are in the root of $\mathcal{B}_{\Pi'}$, we can determine the bounds on floats of activity a, $\underline{f}_a = \max\{0, 9-5-5\} = 0$ and $\overline{f}_a = \max\{0, 12-0-3\} = 9$.

Let us analyse the running time of $Floats$. We implement path trees as in [15], i.e. we use globally biased binary trees [1] to represent path trees . The weight $wt(\delta)$ of node δ of tree $\mathcal{T}$ is defined as

$$wt(\delta) = \begin{cases} 1 & \text{if } \delta \text{ is a leaf of } \mathcal{T}, \\ 1 + \sum\limits_{\substack{\gamma \in children(\delta) \\ (\gamma,\delta) \text{ is dashed}}} wt(\gamma) & \text{otherwise.} \end{cases}$$

Sleator and Tarjan [15] have proved that using partitioning by size (or equivalently by weight) and a representation of path trees as globally biased binary trees, $expose$ and $conceal$ require $\mathcal{O}(\log n)$ time. Hence, operation $Floats$ takes $\mathcal{O}(\log n)$ time.

By the above and using the data structure of [15], we can now formulate our main result.

Theorem 2. *Let G be a series-parallel st-dag. Then there exists a data structure that uses $\mathcal{O}(n)$ space and allows to perform operation $Floats$ in $\mathcal{O}(\log n)$ time.*

From Theorem 2 we obtain the following corollary.

Corollary 1. *Let G be a series-parallel st-dag. Then there exists a data structure that uses $\mathcal{O}(n)$ space and allows to compute bounds on floats of all activities in G in $\mathcal{O}(n \log n)$ time.*

5 Final remarks

In this paper, we have considered the problem of computing bounds on floats of all activities in a network having series-parallel topology with uncertain durations represented by means of interval numbers.

We have proposed an $\mathcal{O}(n)$ time algorithm, based on series and parallel reduction, for computing the bounds on floats of a single activity in the network. Applying this algorithm to each activity in the network leads to a method that takes $\mathcal{O}(n^2)$ times.

We have improved this result by developing an algorithm that runs in $\mathcal{O}(n \log n)$ time and requires $\mathcal{O}(n)$ space for a data structure. Also the algorithm can be used for evaluating the possible and necessary criticality of all activities in the network, since an activity $\omega \in A$ is possibly (resp. necessarily) critical if and only if $\underline{f}_\omega = 0$ (resp. $\overline{f}_\omega = 0$). Making use of results given in [9], it would be interesting to find an $\mathcal{O}(n \log n)$ algorithm, similar in spirit to that presented here, for computing the bounds on latest starting times of all activities in the network.

Our algorithms (in particular the algorithm of Section 4.2) may be used for solving problems in a series-parallel network with activity duration times given in the form of fuzzy numbers, when they are stated in the framework of *possibility theory* (see [4], [5], [9]). That is, the problems of computing fuzzy floats of all the activities and their possible and necessary criticality degrees. This follows from the fact that every fuzzy number can be decomposed into a family of intervals according to its level-cuts.

References

[1] S. W. Bent, D.D. Sleator, R. E. Tarjan, Biased Search Trees, *SIAM Journal on Computing* 14 (1985) 545–568.

[2] R.F. Cohen, R. Tamassia, Dynamic Expression Trees, *Algorithmica* 13 (1995) 245–265.

[3] S. Chanas, J. Kamburowski, The use of fuzzy variables in PERT, *Fuzzy Sets and Systems* 5 (1981) 1–19.

[4] S. Chanas, D. Dubois, P. Zieliński, On the sure criticality of tasks in activity networks with imprecise durations, *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics* 32 (2002) 393–407.

[5] S. Chanas, P. Zieliński, Critical path analysis in the network with fuzzy activity times, *Fuzzy Sets and Systems* 122 (2001) 195–204.

[6] S. Chanas, P. Zieliński, The computational complexity of the criticality problems in a network with interval activity times, *European Journal of Operational Research* 136 (3) (2002) 541–550.

[7] S. Chanas, P. Zieliński, On the hardness of evaluating criticality of activities in a planar network with duration intervals, *Operations Research Letters* 31 (2003) 53-59.

[8] D. Dubois, H. Fargier, V. Galvagnon, On latest starting times and floats in activity networks with ill-known durations, *European Journal of Operational Research* 147 (2003) 266–280.

[9] H. Fargier, V. Galvagnon, D. Dubois, Fuzzy PERT in series-parallel graphs, *9-th IEEE Int. Conf. on Fuzzy Systems*, San Antonio, TX, 2000 717–722.

[10] M. Hapke, A. Jaszkiewicz, R. Słowiński, Fuzzy project scheduling system for software development, *Fuzzy Sets and Systems* 67 (1994) 101–117.

[11] J.E. Kelley, M.R. Walker, Critical path planning and and Scheduling, in: *Proceedings of the Eastern Joint Computational Conference* 16 (1959) 160–172.

[12] F.A. Loostma, *Fuzzy Logic for Planning and Decision-Making*, Dordrecht: Kluwer Acad. Publ., 1997.

[13] H. Prade, Using fuzzy sets theory in a scheduling problem: a case study, *Fuzzy Sets and Systems* 2 (1979) 153–165.

[14] H. Rommelfanger, Network analysis and information flow in fuzzy environment, *Fuzzy Sets and Systems* 67 (1994) 119–128.

[15] D.D. Sleator, R. E. Tarjan, A Data Structure for Dynamic Trees, *Journal of Computer and System Science* 26 (1983) 362–391.

[16] J. Valdes, R. E. Tarjan, E. L. Lawler, The Recognition of Series Parallel Digraphs, *SIAM Journal on Computing* 11 (1982) 298-313.

[17] P. Zieliński, Latest starting times and floats of activities in networks with uncertain durations, *Proceedings of An International Conference in Fuzzy Logic and Technology*, Eusflat 2003, 10-12 September 2003, Zittau, Germany, 586-591.