Iranian Journal of Operations Research
Vol. 4, No. 1, 2013, pp. 39-54

An Algorithm to Obtain Possibly Critical Paths in Imprecise Project Networks

R. Morovatdar[1], A. Aghaie[*,2], E. Roghanian[3], A. Asl Haddad[4]

We consider criticality in project networks having imprecise activity duration times. It is well known that finding all possibly critical paths of an imprecise project network is an NP-hard problem. Here, based on a method for finding critical paths of crisp networks by using only the forward recursion of critical path method, for the first time an algorithm is proposed which can find all possibly critical paths of interval-valued project networks. The proposed algorithm considers interactivity among paths which has not been yet considered in the fuzzy project scheduling literature. The extension of the proposed algorithm to the fuzzy network calculates criticality degrees of activities and paths of projects without any need to enumerate all project paths. Although algorithms for calculating criticality degrees in fuzzy networks have been previously proposed, despite the fact that they mostly consider a specific type of fuzzy numbers as activity duration times, the exiting algorithms do not discriminate possibly critical paths before calculating the criticality degrees. The computational experience on a series of well-known project samples confirms the algorithm to be remarkably more efficient than similar algorithms for fuzzy networks.

Keywords: *Project scheduling, Fuzzy PERT, Critical path, Imprecise network.*

Manuscript received on 23/1/2013 and accepted for publication on 8/5/2013.

1. Introduction

Almost every project manager wants his project to be finished on time, and identification of project bottlenecks plays a major role for this purpose. A typical project consists of activities which are related to rational or physical precedence relations. In other words, activities assembled by precedence relations form a project network.

Bottlenecks of projects are actually the critical paths of these projects. A critical path is the longest path from the start (the first event) of the project to the end (the last event) of the project. The project critical path determines the smallest feasible makespan for the project. Activities and events of a critical path are called critical activities and critical events respectively. The first attempt to find critical paths of a project was made by Kelly and Walker [14] who proposed a method called the Critical Path Method (CPM). This method, based on a series of forward and backward computations, calculates the earliest time that an activity can start or finish in a way that all its predecessors are finished, and the latest time that an activity can finish or start in a way to not result in a delay for the project. The earliest start (finish) time of a critical activity is the same as its latest

* Corresponding Author.
[1] Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran. E-mail: morovatdar@dena.kntu.ac.ir
[2] Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran. E-mail:aaghie@kntu.ac.ir
[3] Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran. E-mail:e_roghanian@kntu.ac.ir
[4] Department of Industrial Engineering, K.N. Toosi University of Technology, Tehran, Iran. E-mail:ahaddad@kntu.ac.ir

start (finish) time. In other words, critical activities should start and finish in an exact point of time, since otherwise the project may not meet its due time. Kelly [13] also exploited a linear mathematical model to search for critical paths of a project. We can distinguish, using CPM or solving a linear model, all critical activities, all critical events, and all critical paths of a project network in a polynomial of time with respect to the network size.

The assumption that all project network characteristics are known and deterministic is the main problem of CPM. There are too many factors affecting project characteristics which are not under the control of the project organization. For example, activity duration times, as the main characteristic of activities, cannot be estimated exactly. This problem was promptly considered by Malcolm et al. [18] who proposed the Project Evaluation and Review Technique (PERT). In PERT, each project activity duration time is shown as a random variable with a probability distribution. However, PERT has many restricting assumptions, and many researchers have criticized it; see Kotiah and Wallace [16].

Activity duration times are usually estimated by experts in imprecise and vague values. Thaus, so treating uncertainty of project activities by fuzzy or interval theory instead of probability theory was recommended in the literature (Herroelen and Leus [12]). At first, Chanas and Kamburowski [2] used fuzzy theory for the project scheduling problem. The concept of fuzzy scheduling was expanded then in many ways like considering generalized precedence relations (Yakhchali and Ghodsypour [22]) or considering an imprecise structure for the project network (Morovatdar et al. [19]). Fuzzy scheduling methods were the basis for several following researches on criticality (Chen and Huang [8], Chen [7], and Yakhchali and Ghodsypour [23]). Definition for fuzzy criticality (*f*-criticality) was originally proposed by Chanas and Zielinski [3]. They also proposed two methods for finding criticality degree of each *f*-critical path. They claimed the problem of finding criticality degree of an *f*-critical activity or an *f*-critical event to be NP-hard. Chanas and Zielinski [4] simplified the uncertainty of activities from fuzzy duration times to interval duration times. They defined interval criticality (*i*-criticality) for project activities, events and paths. They solved the problem of determining an arbitrary *i*-critical path and the problem of asserting the *i*-criticality of a fixed path, in a polynomial time with respect to the size of the network. They also proved that the problem of asserting the *i*-criticality of a fixed activity or a fixed event and the problem of finding *k* *i*-critical paths in an interval-valued network were NP-hard. Chanas et al. [5] integrated the concept of *i*-criticality and *f*-criticality and presented "possible criticality" for all imprecise project networks. They also proposed three new concepts including necessary noncriticality, necessary criticality, and possible noncriticality to complement the possible criticality notion. Morovatdar et al. [20] expanded the concept to dominant critical paths filling the wide gap between possible criticality and necessary criticality. Chanas and Zielinski [6] proved that even for planar networks with duration time intervals the problem of asserting whether an activity is possibly critical is NP-complete, as well as the problem of computing bounds on the float of an activity being NP-hard. Yakhchali and Ghodsypour [23] proposed an algorithm for the problem of determining types of criticality for all activities in the network with interval durations. A path enumeration approach for the analysis of critical activities was also presented by Yakhchali [24] for fuzzy project networks.

Despite much research an criticality in imprecise project networks, finding all critical paths of the interval-valued networks have not yet been addressed in the literature,. Moreover, all proposed methods in the fuzzy project networks (Chanas and Zielinski [3], Chen [7], and Yakhchali [24]) are based on enumerating all the project paths, and attempt to discriminate possibly critical paths before calculating related criticality degrees has yet to be made. This paper attempts to determine all possibly critical paths in interval-valued project networks, and by extending the concept to the fuzzy project networks, a very practical and time efficient algorithm is proposed.

In the next section, a brief introduction to the criticality concept in crisp and imprecise networks is presented. Section 3 demonstrates an algorithm for finding critical paths of crisp project networks by only using the forward recursion of CPM. Based on the algorithm for crisp networks, a new algorithm is developed in Section 4 to find all possibly critical paths of an interval-valued project network. For this purpose, the concept of interactivity among project paths is discussed. By means of bisection of possibility intervals, Section 5 extends the proposed algorithm to the fuzzy project networks. Finally, the performance of the algorithm is evaluated on real-world project samples, and a brief comparison is made between the new algorithm and the proposed algorithm by Yakhchali [24].

2. Criticality in Project Networks

We review CPM after presenting the concept of project networks. A project network can be defined as a directed acyclic graph $G=(V,E)$, where V is the set of graph vertices, that is a $1\times n$ array, giving project events, and $E \in V \times V$ is the set of graph edges, that is an $n\times n$ matrix, giving project activities. This network is generally called Activity Arc (AOA) network. Without loss of generality, we assume that network events (vertices) are numbered in successive order so that every network activity (arc) leads from a lower to a higher event number; therefore, matrix E is an upper triangular matrix. In addition, there is always a source event in the network with the first number which is not the finish vertex of any activity, and there is always a sink event with the last number which is not the start vertex of any activity (Demeulemeester and Herroelen [9]).

CPM tries to find the earliest and the latest times for project network activities and events. The earliest occurrence time of project events can be calculated by the following forward recursion formula (Kelley and Walker [14]):

$$T_i = \begin{cases} 0, & \text{for } i = 1 \\ \max\limits_{h \in Pred_i} T_h + d_{hi}, & \text{for } i = 2, \dots, n, \end{cases} \tag{1}$$

where T_i is the earliest occurrence time of the ith event, d_{hi} is the (h, i) activity duration time, and $Pred_i = \{h \in V | (h, i) \in E\}$ is the set of the ith event predecessors. The latest occurrence time of project events can be also obtained by the following backward recursion formula:

$$L_i = \begin{cases} T_n, & \text{for } i = n \\ \min\limits_{h \in Succ_i} L_h - d_{ih} & \text{for } i = 1, \dots, n-1, \end{cases} \tag{2}$$

where L_i is the latest occurrence time of the ith event, and $Succ_i = \{h \in V | (i,h) \in E\}$ is the set of the ith event successors. By means of these two times, the slack times of event i and activity $(i\ ,j)$ can be calculated respectively by the following formulas:

$$F_i = L_i - T_i \tag{3}$$
$$F_{i-j} = L_j - T_i - d_{ij}. \tag{4}$$

Generally, in the literature there are two pairs of definitions for criticality. The first pair which is described as definitions 1 and 2 below is more well-known and is the basis for CPM (Kelley and Walker [14]).

Definition 1: An activity (event) is critical if and only if its slack time is zero.

Definition 2: A path is critical if and only if all activities belonging to it are critical.

The pitfall of these definitions show up when the project is makespan is less than the project's due date; this means all the project activities have a slack time greater than zero. In this case, based on the above definitions, the project network does not have any critical activity or path. Many researchers (e.g., Kuchta [17], and Zammori et al. [25]) have tried to vary the above definitions in order to find critical paths of project networks, in this situation. On the other hand, the advantage these definitions is due to the existence of easy methods (e.g., CPM, PERT, etc.) for finding critical activities and critical paths.
The second pair of definitions is as follows (Demeulemeester and Herroelen [9]).

Definition 3: A path is critical if and only if it is the longest path in the network.

Definition 4: An activity (event) is critical if and only if it belongs to a critical path.

Based on these definitions, and regardless of the project's due time, we can always find critical paths and critical activities of the projects. In fact, Definition 3 is the opposite of the shortest path definition and it can be used to extend methods for solving the shortest path problem to the critical path problem. For determining criticality of an activity, however, all critical paths of the project should be determined first. But, since there may be an exponential number of critical paths with respect to the number of events (n), using Definition 4, there is no way to find all critical activities of a project in a time upper bounded by a polynomial order of n.

In light of the above, definitions 3 and 4 for finding critical paths and critical activities may not be applicable in real projects, and so definitions 1 and 2 are used more often. In the following sections we will show that definitions 3 and 4 can be more useful in special situations.

The concept of configuration Ω (Buckley [1]) can be used to extend the definition of criticality to imprecise networks. A configuration is a set of deterministic values, d_{ij}, for imprecise (interval) activity duration times, $\tilde{d}_{ij}$, so that the relation $d_{ij} \in \left[\underline{d}_{ij}, \overline{d}_{ij}\right]$ holds. The following definitions for possible criticality are extracted from Chanas and Zielinski [4].

Definition 5: A path is possibly critical if and only if it is critical in its usual sense (definitions 2 or 3) in at least one configuration.

Definition 6: An activity (event) is possibly critical if and only if it is critical in its usual sense (definitions 1 or 4) in at least one configuration.

The above two definitions can result in the following two statements, as well (Chanas et al. [5]).

Statement 1: A path is possibly critical if and only if all activities belonging to it are possibly critical.

Statement 2: An activity (event) is possibly critical if and only if it belongs to a possibly critical path.

By considering an arbitrary configuration for an interval-valued project network, and using conventional methods for finding critical paths, we can reach one or some possibly critical paths of a project. On the other hand, we can assert the possible criticality of a fixed path, p, using the following configuration (Chanas and Zielinski [4]):

$$d_{ij} = \begin{cases} \overline{d}_{ij,} & \text{if } (i,j) \in p \\ \underline{d}_{ij}, & \text{if } (i,j) \notin p. \end{cases} \tag{5}$$

Although asserting the possible criticality of each path is easy, Chanas and Zielinski [4] proved that finding k possibly critical paths in a project network is an NP-hard problem. In the following section, a heuristic method for finding all possibly critical paths of imprecise project networks is presented.

In fuzzy networks, instead of labeling a path as critical or non-critical, the possibility that the path belongs to the critical path set is calculated. The possibility distribution of configuration Ω of activity duration times is $\pi(\Omega) = \min_{(i-j)\in V} \mu_{\Omega}(\tilde{d}_{ij})$. The following formula determines the possibility that a path is critical (Chanas and Zielinski [3]):

$$\text{Poss}(p \text{ is critical}) = \sup_{\Omega^*:\, p \text{ is critical}} \pi(\Omega^*). \tag{6}$$

The above possibility is usually called the criticality degree of path p. The criticality degree of activities can also be obtained by a similar formula. Although we can nominate all the paths with criticality degree of more than zero as possibly critical paths, the uncertainty embedded in the fuzzy numbers would be disregarded if it is not stated with the criticality degree.

3. Critical Paths in Crisp Project Networks

Using the concept put forward in definition 3, we can distinguish critical paths for deterministic networks by only calculating the forward recursion of CPM. Algorithm 1, below based on Dijkstra's Algorithm (Dijkstra [10]) for finding the shortest path(s) of a network, easily provides a

method. Let us denote the set of longest paths from the source event up to *i*th event by P_i, $P_i = \{P_i^l | l = 1, \dots, |P_i|\}$.

Algorithm 1: Compute the set of all critical paths.
Require: Project network $G = (V, E)$ and Activity durations D.
Ensure: Set of all critical paths (P_n).
1: $T_1 \leftarrow 0$;
2: for i : 2 to n
3: $T_i \leftarrow \max\{T_h + d_{hi} | h \in Pred_i\}$;
4: $Pred_i^* \leftarrow \{h | (h \in Pred_i)$ AND $(T_i = T_h + d_{hi})\}$;
5: next i
6: $V' \leftarrow \{n\}$;
7: for i : n to 2
8: if $i \in V'$ then $V' \leftarrow V' \cup Pred_i^*$
9: next i
10: $P_1 = \{P_1^1\} \leftarrow \{\{1\}\}$;
11: for i : 2 to n
12: if $i \notin V'$ then next i
13: $m \leftarrow 0$;
14: for $h \in Pred_i^*$
15: for l : 1 to $|P_h|$
16: $m \leftarrow m + 1$;
17: $P_i^m \leftarrow P_h^l \cup \{i\}$;
18: next l
19: next h
20: next i

In Algorithm 1, $Pred_i^*$ is the set of active predecessors realizing the *i*th event, and V' is the set of critical events. This algorithm is based on the forward recursion of CPM. But, besides calculating the early occurrence times of events, Algorithm 1 discriminates the predecessor events and determines the active predecessor events. Then, it distinguishes the project critical events, and by using these critical events, the algorithm determines the longest paths from the source event to the *i*th (critical) event by considering all combinations of i and the longest paths to the active predecessors of event i. Accordingly, the number of longest paths to each event can be calculated as $|P_i| = \sum_{h \in Pred_i^*} |P_h|$.

For a better presentation of Algorithm 1, a flowchart is given in Fig. 1.

Note that above each event (node) in Fig. 2, both the longest path(s) from the first event to that event and the length of the path(s) are labeled as P_i: T_i . It is obvious that the length of the longest path to each event is equal to the earliest occurrence time of that event. Only for event 6, the longest path is not distinguished, because event 6 does not belong to the critical event set of the project. The sink event label (event 7) shows paths {1,2,5,7} and {1,3,4,5,7} which are critical paths of length 32.

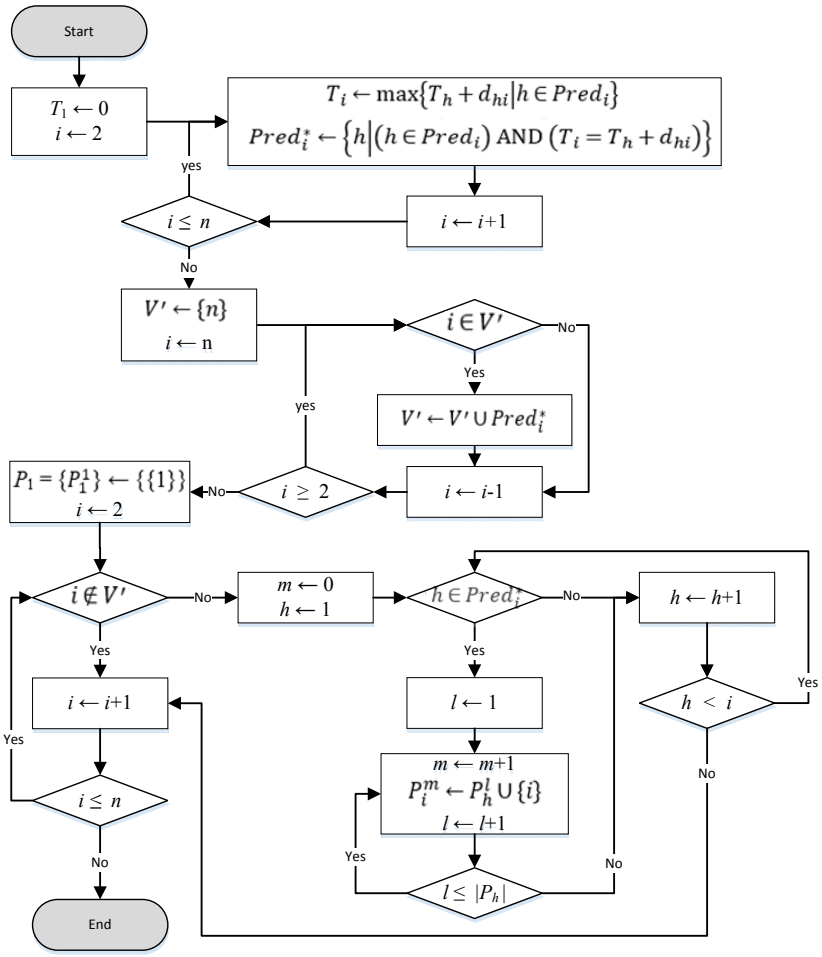

Figure 1. Flowchart of Algorithm 1

A simple example for application of this algorithm is presented in Fig. 2.

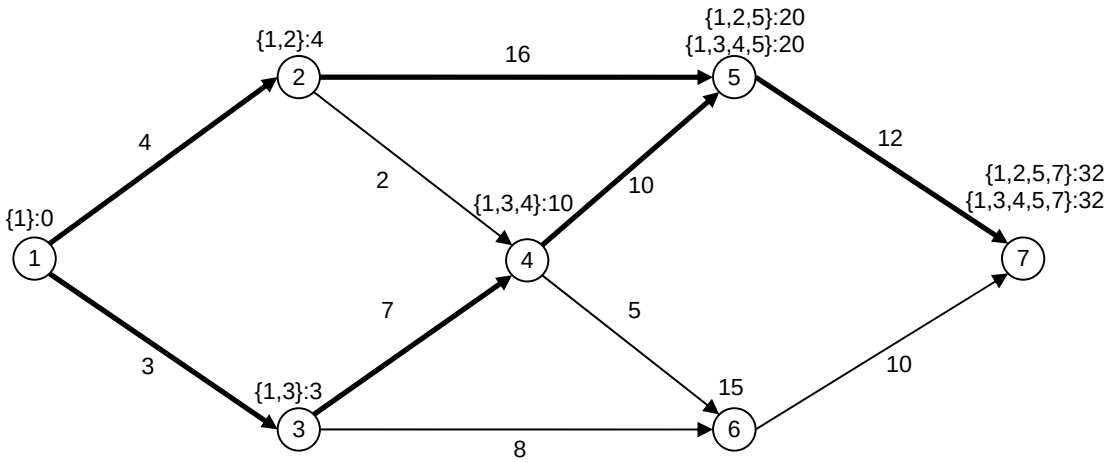

Figure 2. Critical paths of a deterministic network

4. Critical Paths in Interval-Valued Project Networks

Algorithm 1 can be extended to imprecise networks. All possibly critical paths of a network with interval activity duration times can also be distinguished by using the forward recursion of CPM. But, let us first figure out the concept of active predecessor in the case of imprecise activity duration times. As implied before, the earliest occurrence time of event i in a crisp network can be distinguished only by active predecessors. In a imprecise network, those predecessor events which have the possibility to determine the earliest occurrence time of event i can be nominated for the active predecessor events. For $h \in Pred_i$ to be an active predecessor event, interval $\tilde{T}_h \oplus \tilde{d}_{hi}$ should have intersection with interval $\tilde{T}_i$, or $\underline{T}_i \leq \overline{T}_h + \overline{d}_{hi}$ and $\overline{T}_i \geq \underline{T}_h + \underline{d}_{hi}$, with the latter always being true. It should be noted that for determining the active predecessors, interactivity among activities is not considered. Interactivity concept will be discussed more in this section.

Let us denote $t_i = \{\tilde{t}_i^l | l = 1, \ldots, |P_i|\}$ as the set of possibly critical path lengths from the source event to the ith event, where $\tilde{t}_i^l$ is the length of path P_i^l. The following algorithm can find all possibly critical paths in an interval-valued project network.

Note that in Algorithm 2, V' is the set of potential possibly critical events, and $U \setminus A$ denotes the set difference of U and A. At line 3, the earliest occurrence times of project events can be calculated by using interval (fuzzy) operators (see Zimmermann [26]).

Algorithm 2 uses only the forward calculations of CPM to find all possibly critical paths. In the forward calculations, interval (fuzzy) operators can be used without encountering any problem like negative times in the backward recursion (Dubois et al. [11]). So, the algorithm does not have the deficiency encountered in most exiting algorithms.

Algorithm 2 is basically similar to Algorithm 1, but it can treat imprecise networks. The main difference between these two algorithms is that in Algorithm 2 each possibly critical path ending up to an active predecessor event of event i can not necessarily construct a possibly critical path for event i. As discussed by Okada [21] for fuzzy networks, the reason is the interactivity among the paths. “Non-interactivity” in fuzzy theory corresponds to “independency” in probability theory. Two paths are called interactive if and only if they share at least one activity having a non-zero duration time (Okada [21]). The example in Fig. 3 reveals the reason why interactivity among two paths may cause one of the paths to be necessarily non-critical, although it seems to be a possibly critical path.

The network of Fig. 3 has two paths P^1 = {1,2,4} and P^2 = {1,2,3,4} with lengths t^1 = [7,11] and t^2 = [5,9], respectively. The lengths of these paths overlap with each other, and so it seems that they are both eligible to be possibly critical. But, with a better look, we can notice that both paths share activity (1,2) which has inflated the path lengths. If we do not consider this activity, the paths will be deduced to {2,4} and {2,3,4}, respectively, with lengths [4,5] and [2,3]. Clearly, path {2,3,4} is shorter than path {2,4} and it cannot be a member of the possibly critical paths set. Hence, {1,2,4}. is only possibly critical path in Fig. 3

In lines 18 to 30 of Algorithm 2, the pairs of interactive paths with one dominating the other are to be found. In this case, the path with the smallest length will be removed from the set of possibly critical paths.

Algorithm 2: Compute the set of all possibly critical paths.
Require: Project network $G = (V, E)$ and activity durations $\widetilde{D}$.
Ensure: Set of all possibly critical paths (P_n).
1: $\tilde{T}_1 \leftarrow [0,0]$;
2: for i : 2 to n
3: $\quad \tilde{T}_i \leftarrow \widetilde{\max}_{h \in Pred_i}\left(\tilde{T}_h \oplus \tilde{d}_{hi}\right)$;
4: $\quad Pred_i^* \leftarrow \left\{h \middle| (h \in Pred_i) \text{ AND } \left(\overline{T}_h + \overline{d}_{hi} \geq \underline{T}_i\right)\right\}$;
5: next i
6: $V' \leftarrow \{n\}$;
7: for i : n to 2
8: $\quad$ if $i \in V'$ then $V' \leftarrow V' \cup Pred_i^*$
9: next i
10: $P_1 \leftarrow \{P_1^1\} \leftarrow \{\{1\}\}$; $t_1 \leftarrow \{\tilde{t}_1^1\} \leftarrow \{[0,0]\}$;
11: for i : 2 to n
12: $\quad$ if $i \notin V'$ then next i
13: $\quad$ for $h \in Pred_i^*$
14: $\quad\quad q \leftarrow |P_i|$; $m \leftarrow 0$;
15: $\quad\quad$ for l : 1 to $|P_h|$
16: $\quad\quad\quad$ if $\overline{t}_h^l + \overline{d}_{hi} < \underline{T}_i$ then next l
17: $\quad\quad\quad$ *flag* ← True;
18: $\quad\quad\quad$ for j : 1 to q
19: $\quad\quad\quad\quad \tilde{c} \leftarrow \tilde{t}_h^l \oplus \tilde{d}_{hi}$; $\tilde{c}' \leftarrow \tilde{t}_i^j$;
20: $\quad\quad\quad\quad$ for (r-s) which is common activity for P_h^l and P_i^j
21: $\quad\quad\quad\quad\quad \tilde{c} \leftarrow \left[\underline{c} - \underline{d}_{rs}, \overline{c} - \overline{d}_{rs}\right]$; $\tilde{c}' \leftarrow \left[\underline{c}' - \underline{d}_{rs}, \overline{c}' - \overline{d}_{rs}\right]$;
22: $\quad\quad\quad\quad$ next (r-s)
23: $\quad\quad\quad\quad$ if $\overline{c} < \underline{c}'$ then
24: $\quad\quad\quad\quad\quad$ *flag* ← False;
25: $\quad\quad\quad\quad\quad$ exit for loop;
26: $\quad\quad\quad\quad$ else if $\overline{c}' < \underline{c}$ then
27: $\quad\quad\quad\quad\quad P_i \leftarrow P_i \backslash \{P_i^j\}$; $t_i \leftarrow t_i \backslash \{\tilde{t}_i^j\}$;
28: $\quad\quad\quad\quad\quad j \leftarrow j - 1$; $q \leftarrow q - 1$;
29: $\quad\quad\quad\quad$ end if
30: $\quad\quad\quad$ next j
31: $\quad\quad\quad$ if *flag* = True then
32: $\quad\quad\quad\quad m \leftarrow m + 1$;
33: $\quad\quad\quad\quad P_i^{q+m} \leftarrow P_h^l \cup \{i\}$; $\tilde{t}_i^{q+m} \leftarrow \tilde{t}_h^l \oplus \tilde{d}_{hi}$;
34: $\quad\quad\quad$ end if
35: $\quad\quad$ next l
36: $\quad$ next h
37: next i

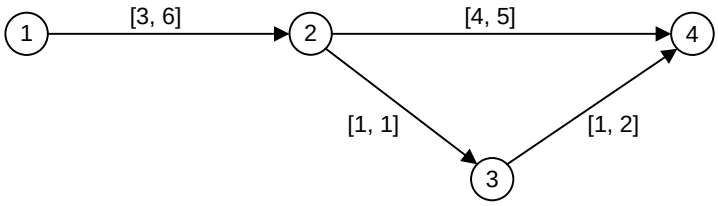

Figure 3. Interactivity among paths of a network

For an illustration of the proposed steps for finding all possibly critical paths project networks, let us consider the network in Fig. 2 with the following activity duration times as given in Table 1, instead of crisp activity duration times shown on the figure.

The 6th event is selected to review the calculation process. Unlike the crisp network, event 6 here is one of the potential critical events of the project network. The earliest occurrence time of event 6 is [9,12] and its active predecessors are {3,4}. Starting from active predecessor 3 and by considering the calculation results for event 3 as (shown in Table 2), the path {1,3,6} with the length [9,12] can easily be added to the set of possibly critical paths of event 6. Considering active predecessor 4, two other paths, namely {1,2,4,6} and {1,3,4,6}, are nominated to be added to the possibly critical path set of this event. The length of path {1,2,4,6} is [6,8], which is clearly smaller than event 6 occurrence time. Hence, as per line 16 of Algorithm 2, we shall discard this path. The length of path {1,3,4,6} is [6,9], having an intersection with the occurrence time of event 6. However, activity (1-3) is common for paths {1,3,6} and {1,3,4,6}, and as per line 21 of Algorithm 2, the deducted paths would be {3,6} with the length [8,10] and {3,4,6} with the length [4,7], with the former being clearly greater than the latter. Therefore, the only possibly critical path up to event 6 is {1,3,6}.

The results of calculations for all the events are given in Table 2. The possibly longest paths of the 7th event are actually the possibly critical paths of the project network.

Table 1. Activity duration times of an interval-valued project network

Activity	1-2	1-3	2-4	2-5	3-4	3-6	4-5	4-6	5-7	6-7
Duration	[1,1]	[1,2]	[2,3]	[6,9]	[1,3]	[8,10]	[1,2]	[3,4]	[4,6]	[0,1]

Table 2. The longest paths from the first event to the *i*th event

Event	$Pred_i$	T_i	$Pred_i^*$	P_i	t_i
1	{}	[0,0]	{}	{{1}}	{[0,0]}
2	{1}	[1,1]	{1}	{{1,2}}	{[1,1]}
3	{1}	[1,2]	{1}	{{1,3}}	{[1,2]}
4	{2,3}	[3,5]	{2,3}	{{1,2,4}, {1,3,4}}	{[3,4], [2,5]}
5	{2,4}	[7,10]	{2,4}	{{1,2,5}, {1,3,4,5}}	{[7,10], [3,7]}
6	{3,4}	[9,12]	{3,4}	{{1,3,6}}	{[9,12]}
7	{5,6}	[11,16]	{5,6}	{{1,2,5,7}, {1,3,4,5,7}, {1,3,6,7}}	{[11,16], [7,13], [9,13]}

5. Critical Paths in Fuzzy Project Networks

Algorithms 1 and 2 can be expanded to fuzzy project networks. In fuzzy networks, the possibility of a path to be a member of the critical path set can be calculated, and it is called criticality degree (Chanas and Zielinski [3]). In fact, instead of nominating a path as critical or non-critical, it is represented by a certain degree of possibility to be critical. Algorithm 3 computes an approximate amount (with the accuracy to be at least ε) for the criticality degree of each project path and each project activity. This algorithm can be used for all types of fuzzy numbers as activity duration times. Let us denote $\alpha(i)$ as the possibility level for the jth α-cut, or $\alpha(i)$-cut, on activity duration times, and $P(i)$ as the set of all possibly critical paths related to the interval-valued project network of the jth α-cut. The following lemma should be proved before presenting Algorithm 3.

Lemma 1: If $\alpha(i) < \alpha(i+1)$ then $P(i+1) \subseteq P(i)$.

Proof. Let us suppose there is a path $P^* \in P(i+1)$ with $P^* \notin P(i)$. Considering Definition 5, there exists at least one configuration Ω^*, $d_{rs}^* \in \left[\underline{d}_{rs}^{\alpha(i+1)}, \overline{d}_{rs}^{\alpha(i+1)}\right]$, for which P^* is critical in the usual sense. The membership functions of activity duration times are fuzzy numbers and convex, and thus considering the assumption of Lemma 1 that $\alpha(i) < \alpha(i+1)$, we have $\underline{d}_{rs}^{\alpha(i)} \leq \underline{d}_{rs}^{\alpha(i+1)}$ and $\overline{d}_{rs}^{\alpha(i)} \geq \overline{d}_{rs}^{\alpha(i+1)}$. This means $d_{rs}^* \in \left[\underline{d}_{rs}^{\alpha(i)}, \overline{d}_{rs}^{\alpha(i)}\right]$, and the configuration Ω^* is a valid configuration for interval valued network of $\alpha(i)$-cut. Hence, P^* is a possibly critical path of the network of $\alpha(i)$-cut, which contradicts the first assumption. □

The basis of Algorithm 3 is bisection of the possibility intervals $[\alpha(i), \alpha(i+1)]$ up to the level that there is no difference between $P(i)$ and $P(i+1)$ or the length of the interval is less than ε. In fact, the two sets of possibly critical paths, $P(i)$ and $P(i+1)$, are considered in each iteration for the lower bound and the upper bound of the relative possibility interval. From Lemma 1, all paths being members of $P(i+1)$ are members of $P(i)$ as well, and their minimum criticality degree is $\alpha(i)$. The difference set between $P(i)$ and $P(i+1)$ is noted by the algorithm. The criticality degrees of those paths belonging to the difference set are somewhere in the possibility interval $[\alpha(i), \alpha(i+1)]$. Therefore, by narrowing the interval, the minimum and the maximum limits of the possibility for each path to be critical is found.

By a direct results of Lemma 1, the possibly critical path set of the interval-valued project network made by up the ε-cut of activity duration times, $P(1)$, includes all possibly critical paths of the fuzzy project network with criticality degrees more than ε.

Procedure 1 is a sub-algorithm similar to Algorithm 2. In fact, Procedure 1 can be replaced by Algorithm 2 in line 9 of the algorithm, but Procedure 1 eliminates some extra calculations in this specific case, and makes Algorithm 3 to be more time efficient.

Algorithm 3: Compute the set of criticality degrees of possibly critical paths and the set of criticality degrees of activities.
Require: Project network $G = (V, E)$ and activity durations $\widetilde{D}$.
Ensure: Set of criticality degrees of possibly critical paths (DP) and set of criticality degrees of activities (DA).
1: $\alpha(1) \leftarrow \varepsilon$; $\alpha(2) \leftarrow 1$;
2: $P(1) \leftarrow$ Algorithm 2 ($\tilde{d}_{ij}^{\varepsilon}$);
3: $P(2) \leftarrow$ Algorithm 2 ($\tilde{d}_{ij}^{1}$);
4: do while *flag* = False
5: *flag* ← True;
6: for i : 1 to $|\alpha|$ - 1
7: if $\alpha(i+1) - \alpha(i) > \varepsilon$ and $|P(i+1)| \neq |P(i)|$ then
8: insert $\{ \lambda \leftarrow (\alpha(i) + \alpha(i+1)) / 2\}$ between $\alpha(i)$ and $\alpha(i+1)$;
9: insert {Procedure 1 (i)} between $P(i)$ and $P(i+1)$;
11: *flag* ← False;
12: end if
13: next i
14: end do
15: for i : 1 to $|P(1)|$
16: $DP^i = \max \{\alpha(j) \mid P^i(1) \in P(j)\}$;
17: next i
18: for $(r\text{-}s) \in E$
19: $DA^{rs} = \max \{DP^j \mid (r\text{-}s) \in P^j(1)\}$;
20: next k

Procedure 1: Compute the set of possibly critical paths.
Require: The index i for determining $P(i)$, $P(i+1)$, λ.
Ensure: PS (set of possibly critical paths at λ).
1: $PS \leftarrow P(i+1)$; $PD \leftarrow P(i) \setminus P(i+1)$;
2: for j : 1 to $|PS|$
3: $\tilde{L}_{PS}^{j} \leftarrow \sum_{(r-s)\in PS^j} \tilde{d}_{rs}^{\lambda}$;
4: next j
6: for l : 1 to $|PD|$
7: $\tilde{L}_{PD}^{l} \leftarrow \sum_{(r-s)\in PD^l} \tilde{d}_{rs}^{\lambda}$;
8: *flag* ← True ;
9: for j : 1 to $|PS|$
10: $\tilde{c} \leftarrow \tilde{L}_{PD}^{l}$; $\tilde{c}' \leftarrow \tilde{L}_{PS}^{j}$;
11: for $(r\text{-}s)$ which is common activity for PD^l and PS^j
12: $\tilde{c} \leftarrow \left[\underline{c} - \underline{d}_{rs}^{\lambda}, \overline{c} - \overline{d}_{rs}^{\lambda}\right]$; $\tilde{c}' \leftarrow \left[\underline{c}' - \underline{d}_{rs}^{\lambda}, \overline{c}' - \overline{d}_{rs}^{\lambda}\right]$;
13: next $(r\text{-}s)$
14: if $\overline{c} < \underline{c}'$ then
15: *flag* ← False;
16: exit for loop;
17: end if
18: next j
19: if *flag* = True then
20: $PS \leftarrow PS \cup PD^l$; $L_{PS} \leftarrow L_{PS} \cup L_{PD}^{l}$;
21: end if

22: next l

where $\tilde{d}_{rs}^{\lambda}$ is the λ-cut of the membership function of activity r-s duration time.

6. Computational Experiment

The performance of Algorithm 3 is tested on the real-world sample project networks proposed by Kolisch and Sprecher [15]. The calculations are done for project networks with 30, 60, 90 and 120 activities, each having 90 project network instances. These instances along with the standard project generator (ProGen) can be downloaded from the PSPLIB web site (http://129.187.106.231/psplib/).

The activity duration times of sample project networks are crisp numbers, and to be able to use them in Algorithm 3 they should be converted to fuzzy numbers. The result of Algorithm 3 depends on the uncertainty range of activity duration times, and therefore the uncertainty factor of conversion, δ, should be considered as a key parameter. A project network with a wide range of uncertainty in activity duration times can result in more possibly critical paths and more time for calculating criticality degrees. Here, the crisp number d is converted to a random trapezoidal fuzzy number $[d - \alpha_1, d, \beta d, \beta d + \alpha_2]$, where $\beta \in [1, 1 + \delta]$, $\alpha_1, \alpha_2 \in [0, \delta d]$ and $\delta > 0$.

Algorithm 3 has been executed on a PC with 2.8 GHz AMD dual core processor and 2 GB of Ram. Table 3 demonstrates the average time elapsed for Algorithm 3 to calculate all criticality degrees with different accuracies in results and different number of activities (the uncertainty factor is considered to be δ = 0.5). All the computing times are less than one second, and even for 120 activities with 10^{-6} accuracy, the required execution time of the algorithm is quite negligible.

Fig. 4 shows the results of Table 3 in a chart view for a better illustration of the execution time trends. Although Algorithm 3 is not a polynomial time algorithm, the required execution times confirm that the algorithm can be very practical in real project scheduling, and even by increasing the accuracy, the efficiency of the algorithm still remain acceptable.

Table 3. The computing times of Algorithm 3 for different accuracies (δ = 0.5)

No. of activities		30	60	90	120
No. of tested project networks		90	90	90	90
Average no. of paths		21.07	42.02	61.79	81.48
Average no. of critical paths		7.67	13.29	18.53	25.31
Average time elapsed (milliseconds)	$\varepsilon = 10^{-3}$	0.83	2.83	6.43	12.86
	$\varepsilon = 10^{-4}$	1.10	3.94	8.77	17.30
	$\varepsilon = 10^{-5}$	1.40	5.14	11.12	21.06
	$\varepsilon = 10^{-6}$	1.51	5.93	13.46	24.16

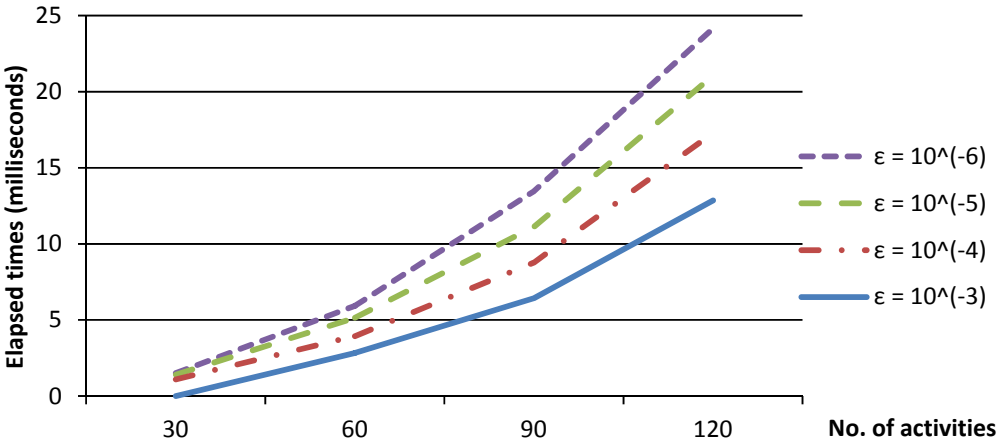

Figure 4. The execution times of Algorithm 3

Table 4 also demonstrates the sensitivity of the algorithm to the uncertainty factor of activity duration times.

Based on the execution times expressed in Table 4, Algorithm 3 is also efficient for different uncertainty factors. In theory, δ can have values more than one, but it should be noted that for $\delta = 1$ the length of the fuzzy number made up for the crisp number d can be up to $3d$ and on the average $1.5d$, which barely happens in real estimations of activity duration times.

The comparison between Algorithm 3 and the proposed algorithm by Yakhchali [24] reveals that there is a large difference between the efficiencies of these two algorithms. Yakhchali's algorithm is not dependent on the membership function of activity duration times, but comparing with the worst case, it is drastically less efficient than Algorithm 3. It should also be noted that the difference between the required computing times of the two algorithms is widened by increasing the network size, that is, project activities.

Table 4. The computing times of Algorithm 3 for different uncertainty factors ($\varepsilon = 10^{-6}$)

No. of activities No. of tested project networks		30 90	60 90	90 90	120 90
Average no. of paths		21.07	42.02	61.79	81.48
$\delta = 0.1$	Average no. of critical paths	1.79	2.01	2.29	2.86
	Average time elapsed (millisecond)	0.17	0.34	0.40	0.93
$\delta = 0.5$	Average no. of critical paths	7.67	13.29	18.53	25.31
	Average time elapsed (millisecond)	1.51	5.93	13.46	24.16
$\delta = 1$	Average no. of critical paths	14.06	28.53	41.89	56.24
	Average time elapsed (millisecond)	4.34	20.77	46.97	88.63
Yakhchali [24] Average time elapsed (millisecond)		119.91	607.24	1743.29	3346.42

7. Concluding Remarks

Most researchers have only considered the difference between the earliest and the latest times of events for finding critical paths and critical activities. Here, based on an algorithm constructed from a crisp shortest path algorithm, a new approach was proposed for imprecise project networks to determine critical paths of projects by using only earliest occurrence times of events. An algorithm was presented to find all possibly critical paths of interval-valued networks. Although the algorithm is not a polynomial time algorithm, it was shown to be quite efficient for real projects to assist the project managers in finding bottlenecks of their projects. To avoid undesirable results, the concept of interactivity among project paths was introduced and monitored in the proposed algorithms. Using the close relations among fuzzy numbers and intervals, the proposed algorithm was extended to handle project networks with fuzzy activity duration times. Instead of asserting a path as critical or non-critical, the proposed algorithm calculates criticality degrees of all project paths along with all project activities. Unlike the exiting algorithms, our proposed algorithm first distinguishes all possibly critical paths of the project and then assigns criticality degrees to them. The efficiency of the algorithm was noted by employing real world project samples. The results showed the algorithm to be efficient, while being quite faster than the similar proposed algorithms for fuzzy project networks. There are usually many possibly critical paths in real applications, and project managers are mostly interested in paths with large criticality degrees. Therefore, developing a more efficient algorithm to find the k most possibly critical paths of project networks turns to be of interest in future research.

References

[1] Buckley, J.J. (1989), Fuzzy PERT, In: G.W. Evans, W. Karwowski and M. Wilhelm (Eds.), Application of Fuzzy Set Methodologies in Industrial Engineering, Elsevier, Amsterdam, pp. 103-125.

[2] Chanas, S. and Kamburowski, J. (1981), The use of fuzzy variables in PERT, *Fuzzy Sets and Systems*, 5, 11-19.

[3] Chanas, S. and Zielinski, P. (2001), Critical path analysis in a network with fuzzy activity times, *Fuzzy Sets and Systems*, 122(2), 195-204.

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

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

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

[7] Chen, S.P. (2007), Analysis of critical paths in a project network with fuzzy activity times, *European Journal of Operational Research*, 183, 442-459.

[8] Chen, C. T. and Huang, S. F. (2007), Applying fuzzy method for measuring criticality in project network. *Information Sciences*, 177, 2448-2458.

[9] Demeulemeester, E.L. and Herroelen, W.S. (2002), Project Scheduling: A Research Handbook, Kluwer Academic Publishers, Dordrecht.

[10] Dijkstra, E.W. (1959), A note on two problems in connection with graphs, *Numerische Mathematik*, 1, 269-271.

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

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

[13] Kelley, J.E. (1961), Critical path planning and scheduling: Mathematical basis, *Operations Research*, 9, 296-320.

[14] Kelley, J.E. and Walker, M.R. (1959), Critical path planning and scheduling, *Eastern Joint Computer Conference*, 16, 160-172.

[15] Kolische R. and Sprecher, A. (1996), PSPLIB: A project scheduling library, *European Journal of Operational Research*, 96, 205-216.

[16] Kotiah, T.C.T. and Wallace, B.D. (1973), Another look at the PERT assumption, *Management Science*, 20(1), 44-49.

[17] Kuchta, D. (2001), Use of fuzzy numbers in project risk (criticality) assessment, *International Journal of Project Management*, 19, 305-310.

[18] Malcolm, D.G., Rosenbloom, J.H., Clark, C.E. and Fazer, W. (1959), Application of a technique for R&D program evaluation (PERT), *Operations Research*, 7(5), 646-669.

[19] Morovatdar, R., Aghaie, A. and Yakhchali, S.H. (2011), Fuzzy network analysis for projects with high level of risks- uncertainty in Time and Structure, *International Journal of Industrial Engineering & Production Research*, 22(1), 73-82.

[20] Morovatdar, R., Aghaie, A. and Roghanian, E. (2013), On dominancy of critical paths in project networks with fuzzy activity duration times, *Fuzzy Set and Systems*, submitted.

[21] Okada, S. (2004), Fuzzy shortest path problems incorporating interactivity among paths, *Fuzzy Set and Systems*, 142, 335-357.

[22] Yakhchali, S.H. and Ghodsypour, S.H. (2010a), On the latest starting times and criticality of activities in a network with imprecise durations, *Applied Mathematical Modelling*, 34(8), 2044-2058.

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

[24] Yakhchali, S.H. (2012), A path enumeration approach for the analysis of critical activities in fuzzy networks, *Information Sciences*, 204, 23-35.

[25] Zammori, A.Z., Braglia, M. and Frosolini, M. (2009), A fuzzy multi-criteria approach for critical path definition, *International Journal of Project Management*, 27, 278-291.

[26] Zimmermann, H.J. (2001), Fuzzy Set Theory and its Application, 4th edition, Kluwer Academic Publishers, London.