ELSEVIER

European Journal of Operational Research 100 (1997) 122–133

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Critical path in an activity network with time constraints

Yen-Liang Chen [a], Dan Rinks [b,*], Kwei Tang [b]

[a] *Department of Information Management, National Central University, Chung-Li, Taiwan 320, Taiwan*

[b] *Department of Information Systems and Decision Sciences, Louisiana State University, Baton Rouge, LA 70803, USA*

Received 12 October 1995; accepted 22 April 1996

Abstract

An acyclic graph with nonnegative weights and with a unique source and destination is called an activity network. A project comprised of a set of activities and precedence relationships can be represented by an activity network and the mathematical analysis of the network can provide useful information for managing the project. In a traditional activity network, it is assumed that an activity can begin any time after all of its preceding activities have been completed. This assumption does not adequately describe many practical situations, in which some kinds of time constraint are usually associated with an activity. In this paper, we investigate two types of time constraint commonly encountered in project management. The first is the *time-window constraint*, which assumes that an activity can begin its execution only in a specified time interval. The second is the *time-schedule constraint*, which requires that an activity begin only at one of pre-specified beginning times. An efficient, linear time algorithm for finding the longest path (critical path) and for analyzing the flow time of each arc is developed for activity networks with these time constraints. © 1997 Elsevier Science B.V.

Keywords: Activity network; Critical path; Longest path; Float time; PERT; Time-window; Time-schedule

1. Introduction

An activity network is an acyclic graph with nonnegative weights and with a unique source and destination. Here, an arc can be viewed as an activity and the precedence relationships between all of the activities are represented by the topology of the network. An arc leaving from a certain node cannot begin until all of the arcs going into this node have been finished. In this kind of activity network *vis-à-vis* a project, an important problem is to find the longest path, because all of the arcs in this path denote the most critical activities of the project: if the activities on the critical path are delayed, then the entire project will be delayed. With this information, the decision maker can monitor these critical activities more closely and take corrective action, if necessary, to control the planned schedule. Another problem of interest in an activity network is to analyze the flow time of each noncritical arc to find out how much float (slack) the arc has. Arcs with longer float time are more adjustable or flexible because, if necessary, we can delay an arc with slack and transfer resources from this activity to support activities on critical arcs. For details and surveys, see [1].

In one way or another, analysis of an activity network is concerned with scheduling (timing) issues. A considerable body of research in PERT

* Corresponding author.

PII S0377-2217(96)00140-3

and/or CPM focuses on various time factor aspects. For instance, some of the research is concerned with estimating activity time more accurately [3,7,9], while another area of research deals with the stochastic nature of activity time. Basically, the stochastic activity time stream of research assumes that the activity durations are nonnegative random variables, and then use statistical or stochastic techniques to find the critical path and critical activities [1,10,11,15,17]. However, there is some controversy about which path is the most critical in a stochastic activity network [16,17]. Suffice to say, it is not easy to correctly estimate the completion time of a project whenever activity times are stochastic [2,8,10,12,13].

Although many aspects of the time factor have been studied, an obvious and important time factor has been largely ignored. In the traditional model, it is assumed that an arc can begin at any time after all of its preceding arcs are finished. Although this is a reasonable requirement, it does not adequately describe many practical situations. In reality, an activity is seldom ready for execution at *any* time; some kinds of time constraint are usually attached to it.

In this paper, we consider two improvements over the traditional activity network by including two types of time constraint. The first is the *time-window constraint* [6,14], which assumes that an activity can only be started within a specified time interval. In other words, a time-window defines the earliest time and the latest time that an activity can begin its execution after all of its predecessors are completed. The second is the *time-schedule constraint* [4], which assumes that an activity can begin only at one of an ordered schedule of the beginning times. Practical situations of the second kind include, for example, product shipping through scheduled flights, ocean liners, trains, buses or their combinations from one place to another. With a time-schedule constraint, an activity can begin only after all of its predecessors are complete *and* at one of the beginning times in the time schedule.

These two types of constraints are commonly seen in projects. Failure to include time constraints may result in misleading information to the decision maker. For instance, suppose we found the time associated with the longest path is 100 time units when all the time constraints are ignored. We may be fortunate that this is the correct solution because all the time constraints are not binding. It is possible, however, the actual time of this path is much longer than 100 time units, because: 1) some critical arcs are subject to time-windows and the beginning times for them are prior to the time windows; and/or 2) some critical arcs are subject to time-schedules and the beginning times are earlier than the next scheduled times. As a result, this path may not be the critical path because some other paths may have shorter times when time constraints are considered. The solution may even be infeasible, because some critical arcs are subject to time-windows and the beginning times fall outside the time windows. Furthermore, even if the solution remains unaffected when the time constraints are included, the flow time associated with each arc may be seriously misleading. Thus, it is important to include time constraints into activity network analysis.

The remainder of the paper is organized as follows. In Section 2, we formally define the problem and present solution methods for activity networks with time constraints. In Section 3 we discuss the concept of flow time in detail under the time constraints. Conclusions are presented in Section 4.

2. The problem and the solution method

Let $N = (V, A, t, s, d)$ be an activity network, where $G = (V, A)$ is a directed acyclic graph without multiple arcs, $t(u, v) \geq 0$ denotes the processing time of arc (u, v), s is the starting node and d is the destination node. A project, represented by the corresponding network, is completed when every arc, which denotes an activity in the project, is finished. Arcs (activities) are classified into three different categories, i.e., normal arcs, time-schedule arcs and time-window arcs. A normal activity can begin its work at any time after all of its preceding activities are finished. A time-schedule arc (u, v) is subject to the same constraint as a normal arc but with the additional constraint that it is associated with an ordered list of scheduled beginning times,

$$\mathrm{TS}(u, v) = \left(\mathrm{ts}_1(u, v), \mathrm{ts}_2(u, v), \ldots, \mathrm{ts}_n(u, v)\right),$$

such that the arc can begin only at one of the scheduled times. A time-window arc (u, v) is the

same as a normal arc but with the additional constraint that a time interval,

$$I(u,v) = (begin(u,v), end(u,v)),$$

is associated with it such that the activity must begin in this time interval.

In a traditional activity network consisting of only normal arcs, the longest path of a project and float times for activities are useful information for project management. A longest path in network N constitutes the most critical activities of the project, i.e. those activities which if delayed will cause the entire project to be delayed. The length (duration) of this critical path is the minimum time required for finishing the project. An arc's float time $f(u,v)$ is the difference between the arc's latest beginning time and the arc's earliest beginning time. This information tells us how long arc (u,v) can be delayed from its earliest beginning time without increasing the project's duration.

However, if the underlying network consists of normal arcs, time-window arcs and time-schedule arcs, then the above properties no longer hold. Unlike a traditional activity network, the latest beginning times of the arcs on the longest path may be greater than their corresponding earliest beginning times. Thus, an arc on the critical path may have a positive float time, a characteristic normally associated with the arcs not on the critical path in the traditional activity network. In addition, the interpretation of flow time is different when time constraints are present. An arc in the traditional model is associated with a single, nonnegative float time value, while an arc in our model is associated with two types of nonnegative time components. The first is the arc's float time and the second is the arc's forced waiting time. Since the waiting time is 'forced' as a result of the time constraint, the concept of waiting time is entirely different from that of the float time. Waiting time implies cost because we are forced to wait while doing nothing. On the other hand, float time implies a possible benefit because how long an activity can be delayed when the next activity begins is our own discretion. Consequently, the more float time an arc has, the more flexibility we have in scheduling the activity; conversely, more forced waiting time indicates the activity is less flexible.

Fig. 1 shows an activity network with 8 nodes and

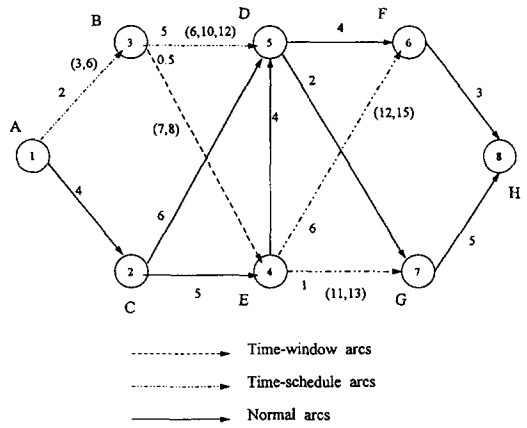

Fig. 1. An example of a scheduling network with three types of arcs.

13 arcs. Activity times are shown as the number on the arc, and the number inside each node is the node's topological order. Arc (A, B) is a time-schedule arc with TS(A, B) = (3, 6); arc (B, D) is a time-schedule arc with TS(B, D) = (6, 10, 12); arc (E, G) is a time-schedule arc with TS(E, G) = (11, 16); arc (E, F) is a time-schedule arc with TS(E, F) = (12, 15); and arc (B, E) is a time-window arc with I(B, E) = (7, 8). Topological order is an order of the nodes in an acyclic graph and this order can be easily constructed in time $O(|A| + |V|)$ by using the algorithms in [5], where $|A|$ and $|V|$ are the numbers of arcs and nodes in the network, respectively. If node u has a smaller topological order than node v, all activities or arcs emanating from node v cannot be predecessors to activities or arcs emanating from node u. Therefore, if we execute all the activities according to their related topological order, then it is ensured that all precedence relationships have been satisfied.

Our algorithm for solving networks with time constraints is similar to the two-phase (forward pass, backward pass) procedure used in the traditional activity networks consisting of only normal arcs. The first phase (forward pass) is to find the earliest beginning time of every node and the longest path from s to d. The second phase (backward pass) is to find the latest beginning time of every node and analyze every arc's float time and waiting time. In the first phase, we examine every node v in the network according to their topological order in as-

cending sequence. For each node v in node set V, a label *early*(v) is assigned, which denotes the earliest beginning time to leave node v. For each arc (u, v) in A, a label *reaching*(u, v) is assigned, which denotes the earliest time to reach node v if we go into node v by arc (u, v). Our algorithm differs from the traditional procedure in the method that *early*(v) is calculated for time-schedule and time-window arcs. In Appendix A, we prove a theorem which validates the following algorithm for finding *early*(v).

Algorithm for the first phase

Step 1. Examine every node v in ascending sequence of topological orders.
Step 1.1. Let *early*$(v) = 0$.
Step 1.2. For each arc (u, v) going into node v,
if (u, v) is normal, then *reaching*$(u, v) =$ *early*$(u) + t(u, v)$
if (u, v) is time-schedule, then
find the smallest scheduled time, say $\text{ts}_k(u, v)$, of arc $(u, v) \geq$ *early*(u)
if no such a $\text{ts}_k(u, v)$ exists, then the problem is infeasible
let *reaching*$(u, v) = \text{ts}_k(u, v) + t(u, v)$.
if (u, v) is time-window, then
if *early*$(u) <$ begin(u, v),
then let *reaching*$(u, v) =$ *begin*$(u, v) + t(u, v)$
else if *begin*$(u, v) \leq$ *early*$(u) \leq$ *end*(u, v),
then let *reaching*$(u, v) =$ *early*$(u) + t(u, v)$
else if *early*$(u) >$ *end*(u, v), then the problem is infeasible
if *reaching*$(u, v) >$ *early*(v), then let *early*$(v) =$ *reaching*(u, v) and $P(v) = u$
Step 2. By reversing from $P(d)$ until $P(s)$, the longest path is found.

Example 1. Using the above algorithm to solve the activity network in Fig. 1, the applying sequence is A, C, B, E, D, F, G, H and the final result is shown in Fig. 2. When we examine node E, we have *early*(A) $= 0$, *early*(C) $= 4$, *early*(B) $= 5$, and *early*(E) $= 0$. There are two arcs, (B, E) and (C, E), going into node E. The first arc, (B, E), is time-window and has

$$\textit{begin}(\text{B}, \text{E}) = 7 > \textit{early}(\text{B}) = 5;$$

hence,

$$\textit{reaching}(\text{B}, \text{E}) = \textit{begin}(\text{B}, \text{E}) + t(\text{B}, \text{E}) = 7 + 0.5 = 7.5,$$

and *early*(E) $= 7.5$. The second arc, (C, E), is a normal arc and has

$$\textit{early}(\text{C}) + t(\text{C}, \text{E}) = 4 + 5 = 9;$$

hence, *early*(E) is updated to 9. Next, we examine node D, which has three arcs going into it. The first arc, (B, D), is time-schedule, its value is

$$\textit{reaching}(\text{B,D}) = \text{ts}_1(\text{B}, \text{D}) + t(\text{B}, \text{D}) = 6 + 5 = 11,$$

and *early*(D) is set to 11. The second arc, (C, D), is normal, its value is

$$\textit{reaching}(\text{C}, \text{D}) = \textit{early}(\text{C}) + t(\text{C}, \text{D}) = 4 + 6 = 10,$$

and *early*(D) is still kept in 11. The last arc, (E, D), is normal, its value is

$$\textit{reaching}(\text{E}, \text{D}) = \textit{early}(\text{E}) + t(\text{E}, \text{D}) = 9 + 4 = 13,$$

and *early*(D) is changed to 13. Since the other arcs are handled in the same way, we omit the details.

Let T_f be the duration of the longest path (e.g., $T_f =$ *early*(d)). We now discuss the second phase of the algorithm. In the backward pass, we examine every node v in this network according to their topological orders in descending sequence. For each node v in node set V, a label *latest*(v) is assigned, which denotes the latest beginning time we can leave node v and still complete all remaining activities in time T_f. After nodes $v_{|V|}, v_{|V|-1}, \ldots, v_{i+1}$ all have been examined, we examine node v_i, where the subscript denotes the topological order value of the node.

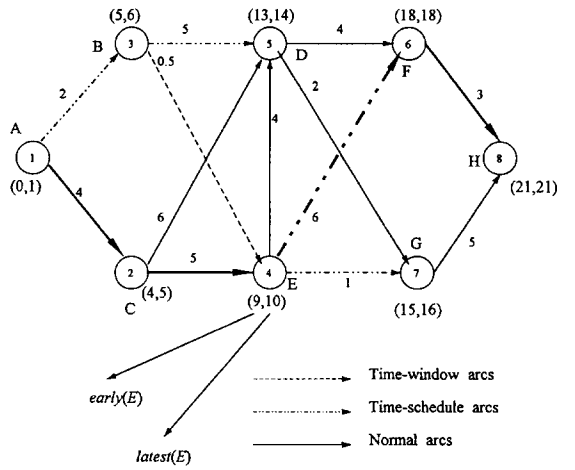

Fig. 2. The first phase for finding the earliest leaving time of each node and the critical path of the network.

The value of $latest(v_i)$ can be defined as

$$\min\{temp(v_i, u) \mid (v_i, u) \in A\},$$

where $temp(v_i, u)$ is the latest time arc (v_i, u) must begin in order to complete all remaining activities in time T_f. Calculations for the backward pass for the three types of arcs are explained as follows.

Case 1: Arc (v_i, u) is normal. Let

$$temp(v_i, u) = latest(u) - t(v_i, u),$$

and let the float time be

$$f(v_i, u) = latest(u) - t(v_i, u) - early(v_i),$$

where both definitions are the same as in the conventional activity network.

Case 2: Arc (v_i, u) is time-schedule. Let

$$temp^{\&}(v_i, u) = latest(u) - t(v_i, u),$$

which is the latest time arc (v_i, u) must begin, if arc (v_i, u) is a normal arc. Because arc (v_i, u) is a time-schedule arc, $temp^{\&}(v_i, u)$ is typically different from $temp(v_i, u)$. Let $ts_k(v_i, u)$ be the largest (latest) scheduled beginning time of arc $(v_i, u) \le temp^{\&}(v_i, u)$. Then, the latest time to begin arc (v_i, u) is $ts_k(v_i, u)$, because beginning at times such as $ts_k(v_i, u)$, $ts_{k-1}(v_i, u), \ldots, ts_1(v_i, u)$, we will arrive at node u sooner than $latest(u)$; but beginning at times such as $ts_{k+1}(v_i, u), ts_{k+2}(v_i, u), \ldots$, we will arrive at node u later than $latest(u)$. Hence, we have $temp(v_i, u) = ts_k(v_i, u)$. We prove (Appendix B, Lemma 1) that it is impossible for $ts_1(v_i, u) > temp^{\&}(v_i, u)$. As a result, we can find a value of k ($k \ge 1$) which meets the condition.

Now, we discuss the float time $f(v_i, u)$ and waiting time on a time-schedule arc (v_i, u). There are two types of waiting time: pre-waiting time $prew(v_i, u)$ and post-waiting time $postw(v_i, u)$. Pre-waiting time is the time that we are forced to wait before beginning the activity of arc (v_i, u) and post-waiting time is the time that we are forced to wait after finishing the activity of arc (v_i, u). The post-waiting time is defined as

$$postw(v_i, u) = latest(u) - ts_k(v_i, u) - t(v_i, u). \tag{1}$$

In the above equation, the latest beginning time of arc (v_i, u) is at time $ts_k(v_i, u)$ and we will finish the work of arc (v_i, u) at time $ts_k(v_i, u) + t(v_i, u)$. So, we will be $latest(u) - ts_k(v_i, u) - t(v_i, u)$ time units sooner than the required latest beginning time of node u.

Next, we will define the pre-waiting time. Let $ts_r(v_i, u)$ be the smallest scheduled beginning time of $(v_i, u) \ge early(v_i)$. Then the earliest possible time to begin arc (v_i, u) is $ts_r(v_i, u)$. This means that even if we are ready to begin work at time $early(v_i)$, we must wait $ts_r(v_i, u) - early(v_i)$ time units before we can proceed to do the work of arc (v_i, u). Hence, we have

$$prew(v_i, u) = ts_r(v_i, u) - early(v_i). \tag{2}$$

The float time $f(v_i, u)$ of arc (v_i, u) can be defined as

$$f(v_i, u) = [latest(u) - early(v_i) - t(v_i, u)] - prew(v_i, u) - postw(v_i, u), \tag{3}$$

where the first term, $[latest(u) - early(v_i) - t(v_i, u)]$, is the traditional definition of float time and $prew(v_i, u)$ and $postw(v_i, u)$ are the pre- and post-waiting times as defined by Eq. (2) and Eq. (1).

Case 3: Arc (v_i, u) is time-window. Because arc (v_i, u) is time-window, $temp^{\&}(v_i, u)$ may be different from $temp(v_i, u)$. If

$$begin(v_i, u) \le temp^{\&}(v_i, u) \le end(v_i, u),$$

then the latest time to begin arc (v_i, u) is the same as $temp^{\&}(v_i, u)$, i.e.,

$$temp(v_i, u) = temp^{\&}(v_i, u).$$

If

$$temp^{\&}(v_i, u) > end(v_i, u),$$

then the latest time to begin arc (v_i, u) is $end(v_i, u)$, i.e.,

$$temp(v_i, u) = end(v_i, u).$$

We prove (Appendix B, Lemma 2) that it is impossible for $begin(v_i, u) > temp^{\&}(v_i, u)$.

Now, we analyze the float time $f(v_i, u)$ and waiting time associated with arc (v_i, u). The following is the definition of post-waiting time:

$$postw(v_i, u) = \max\{0, latest(u) - t(v_i, u) - end(v_i, u)\}. \quad (4)$$

In the above equation, the latest time arc (v_i, u) must begin its work is at time

$$\min\{latest(u) - t(v_i, u), end(v_i, u)\}.$$

If $latest(u) - t(v_i, u)$ is the smaller, then we will arrive node u at time $latest(u)$ and there is no post-waiting time. On the other hand, if $end(v_i, u)$ is the smaller, then we will finish the work of arc (v_i, u) at time $end(v_i, u) + t(v_i, u)$ and this is $latest(u) - t(v_i, u) - end(v_i, u)$ time units sooner than the latest beginning time of node u.

Next, we define the pre-waiting time. If

$$begin(v_i, u) \geq early(v_i),$$

then the earliest possible time to begin arc (v_i, u) is $begin(v_i, u)$. This means that even if we are ready to begin the activity (v_i, u) at $early(v_i)$, we have to wait $begin(v_i, u) - early(v_i)$ time units before we can begin the activity. On the other hand, if

$$begin(v_i, u) < early(v_i) \leq end(v_i, u),$$

then it is not necessary to wait and we can begin the activity at time $early(v_i)$. Finally, it is impossible for $early(v_i) > end(v_i, u)$, because it implies an infeasible solution. Hence, we have

$$prew(v_i, u) = \max\{0, begin(v_i, u) - early(v_i)\}. \quad (5)$$

Similarly, the float time $f(v_i, u)$ of arc (v_i, u) can be defined as

$$f(v_i, u) = [latest(u) - early(v_i) - t(v_i, u)] - prew(v_i, u) - postw(v_i, u). \quad (6)$$

The algorithm for the second phase is given below.

Algorithm for the second phase

Step 1. Let $latest(d) = early(d)$.

Step 2. Examine every node v in descending sequence of topological orders.

Step 2.1. For each arc (v, u) leaving from node v,

if (v, u) is normal, then $temp(v, u) = latest(u) - t(v, u)$

if (v, u) is time-schedule, then

let $temp^{\&}(v, u) = latest(u) - t(v, u)$

let $ts_k(v, u)$ be the largest scheduled time of $(v, u) \leq temp^{\&}(v, u)$

let $temp(v, u) = ts_k(v, u)$

let $ts_r(v, u)$ be the smallest scheduled time of $(v, u) \geq early(v)$.

if (v, u) is time-window, then

let $temp^{\&}(v, u) = latest(u) - t(v, u)$

if $begin(v, u) \leq temp^{\&}(v, u) \leq end(v, u)$, then $temp(v, u) = temp^{\&}(v, u)$

if $temp^{\&}(v, u) > end(v, u)$, then $temp(v, u) = end(v, u)$

Step 2.2. Let $latest(v) = min\{temp(v, u) | (v, u) \in A\}$

Step 3. Examine every node v in descending sequence of topological orders again.

Step 3.1. For each arc (v, u) leaving from node v,

if (v, u) is normal, then

$prew(v, u) = 0$,

$postw(v, u) = 0$,

$float(v, u) = latest(u) - t(v, u) - early(v)$,

if (v, u) is time-schedule, then
$prew(v, u) = ts_r(v, u) - early(v)$,
$postw(v, u) = latest(u) - ts_k(v, u) - t(v, u)$,
and
$float(v, u) = latest(u) - t(v, u) - early(v) - prew(v, u) - postw(v, u)$.
if (v, u) is time-window, then
$prew(v, u) = \max\{begin(v, u) - early(v), 0\}$,
$postw(v, u) = \max\{0, latest(u) - t(v, u) - end(v, u)\}$,
and
$float(v, u) = latest(u) - t(v, u) - early(v) - prew(v, u) - postw(v, u)$.

We show in Appendix B, Lemma 3 that the time complexity of the algorithm is $O(|A| + |V|)$.

Example 2. If the above algorithm is used to solve the activity network in Fig. 2, then the applying sequence is H, G, F, D, E, B, C, A. The latest beginning time information is also indicated in Fig. 2 and all of the other final results are summarized in Table 1.

The results of this example show the following difference of the solution from that obtained from the traditional activity network:

1. The earliest leaving times of the nodes on the critical path may be different from their latest leaving times.
2. The arcs on the critical path may have positive float times. For example, arcs (A, C) and (C, E) have positive float times, although both are on the longest path.
3. The earliest time leaving node A is 0 and the latest time leaving node A is 1, which indicates that the actual time required for this network is 20 time units rather than 21 time units. In other words, the project will be finished at the same time, regardless of whether it begins at time 0 or time 1. This time difference may be called the float time of the system.
4. Arcs may have pre-waiting time and post-waiting time.
5. The possible float time of an arc may be a set of discrete values, not a continuous range. For example, arc (E, G) has $latest(G) = 16$, $early(E) = 9$ and $ts(E, G) = (11, 13)$. Therefore, arc (E, G) has a pre-waiting time of 2 time units, a post-

Table 1
The second phase for computing the pre-waiting time, post-waiting time and float time of each arc

Node	Seq.	Arc	*latest*(*u*)	*early*(*v*)	*t*(*v*, *u*)	*prew*	*postw*	*float*
H	1	(F, H)*[a]	21	18	3			0
H	2	(G, H)	21	15	5			1
G	3	(D, G)	16	13	2			1
G	4	(E, G)	16	9	1	2	2	2
F	5	(D, F)	18	13	4			1
F	6	(E, F)*	18	9	6	3	0	0
D	7	(B, D)	14	5	5	1	3	0
D	8	(C, D)	14	4	6			4
D	9	(E, D)	14	9	4			1
E	10	(B, E)	10	5	0.5	2	1.5	1
E	11	(C, E)*	10	4	5			1
B	12	(A, B)	6	0	2	3	1	0
C	13	(A, C)*	5	0	4			1
A	14							

[a] The arc marked '*' is in the longest path.

waiting time of 2 time units, and a float time of 2 time units. The flow time is either 0 or 2, rather than any value in the range from 0 to 2. In other words, we may either begin this activity at time 11 or 13, but it is not allowed, for example, to begin at time 12.

3. Further discussion of waiting time

An important finding of this paper is the interpretation of flow time when time constraints are considered in an activity network. In this section we provide a more detailed analysis and discussion for flow time as it is decomposed into pre-waiting time and post-waiting time.

To begin, we assume that the arc (v, u) is time-schedule. From previous definitions (1), (2), and (3), we obtain

$$postw(v,u) = latest(u) - ts_k(v,u) - t(v,u),$$

$$prew(v,u) = ts_r(v,u) - early(v),$$

$$float(v,u) = latest(u) - t(v,u) - early(v) - prew(v,u) - postw(v,u).$$

Recall $ts_k(v,u)$ is the largest scheduled time of arc $(v,u) \le latest(u) - t(v,u)$ and $ts_r(v,u)$ is the smallest scheduled time of arc $(v,u) \ge early(v)$.

The above relationships are illustrated in Fig. 3a and contrasted with the traditional flow time. In addition, both pre-waiting time and post-waiting time can be further decomposed into a flexible part and a fixed part (see Fig. 3b and Fig. 3c). First consider the pre-waiting time. Suppose

$$latest(v) < ts_r(v,u).$$

Then the flexible part is the time interval $(early(v), latest(v))$ and the fixed part is the interval $(latest(v), ts_r(v,u))$. We call the interval $(early(v), latest(v))$ flexible because the duration of the waiting time in this time period is at the discretion of the project manager. All activities emanating from node v are ready for execution at the earliest at time $early(v)$ because of precedence relationships, while they must begin no later than at time $latest(v)$ in order to finish the project on schedule. We call the interval $(latest(v), ts_r(v,u))$ fixed because this waiting time period is unavoidable. Although arc (v,u) is ready for execution on or before time $latest(v)$, we still must wait until time $ts_r(v,u)$ because it is the earliest scheduled beginning time for this activity.

Similarly, the post-waiting time can be decomposed into a flexible part and a fixed part. Suppose

$$early(u) > ts_k(v,u) + t(v,u).$$

Then, the fixed part is the time interval $(ts_k(v,u) + t(v,u), early(u))$, and the flexible part is the time interval $(early(u), latest(u))$. The interval $(ts_k(v,u) + t(v,u), early(u))$ is fixed because we have to wait until time $early(u)$ to begin the activities emanating from node u, although the activity of arc (v,u) is finished at time $ts_k(v,u) + t(v,u)$. The interval $(early(u), latest(u))$ is called flexible because we have flexibility to begin the activities leaving from node u at any point in the interval $[early(v), latest(v)]$ without delaying the completion of the project.

Consider the time-schedule arc (E, G) in Fig. 2. We have $early(E) = 9$, $latest(E) = 10$, $t(E,G) = 1$, $ts_r(E,G) = 11$, $ts_k(E,G) = 13$, $early(G) = 15$, and $latest(G) = 16$. Calculations of pre-waiting and post-waiting times and their decomposition into flexible and fixed parts are shown as follows:

The traditional flow time is

$$latest(G) - early(E) - t(E,G) = 16 - 9 - 1 = 6.$$

The pre-waiting time is

$$ts_r(E,G) - early(E) = 11 - 9 = 2.$$

The post-waiting time is

$$latest(G) - ts_k(E,G) - t(E,G) = 16 - 13 - 1 = 2.$$

The float time is

$$latest(G) - t(E,G) - early(E) - prew(E,G) - postw(E,G) = 2.$$

The fixed pre-waiting time is

$$ts_r(E,G) - latest(E) = 11 - 10 = 1.$$

The flexible pre-waiting time is

$$latest(E) - early(E) = 10 - 9 = 1.$$

The fixed post-waiting time is

$$early(G) - ts_k(E,G) - t(E,G) = 15 - 13 - 1 = 1.$$

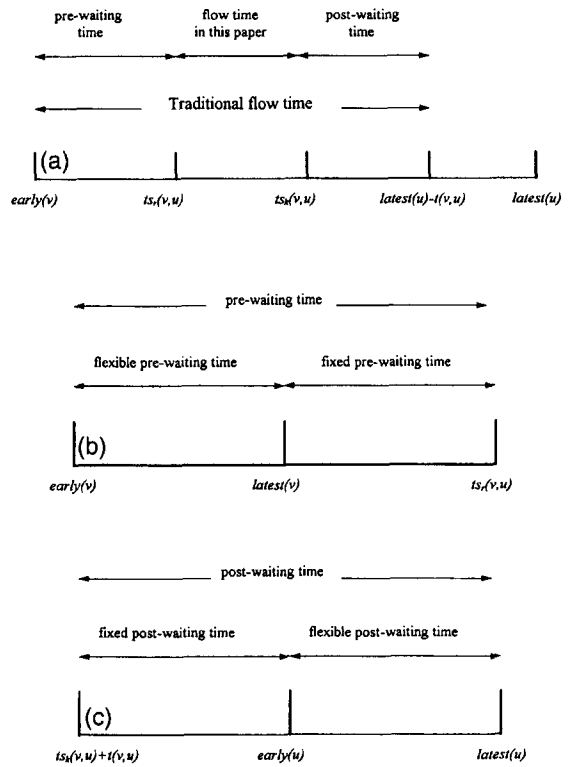

Fig. 3. (a) Different definitions of float time for normal arcs and time-schedule arcs; (b) decomposition of the pre-waiting time into two components; (c) decomposition of the post-waiting time into two components.

The flexible post-waiting time is

$$latest(\mathrm{G}) - early(\mathrm{G}) = 16 - 15 = 1.$$

Next, we consider the case where (v, u) is a time-window arc. Using Eqs. (4), (5), (6),

$$postw(v,u) = \max\{0, latest(u) - t(v,u) - end(v,u)\},$$

$$prew(v,u) = \max\{begin(v,u) - early(v), 0\},$$

and

$$float(v,u) = latest(u) - t(v,u) - early(v) - prew(v,u) - postw(v,u).$$

The above relationships are illustrated in Fig. 4a and compared with the traditional flow time. Fig. 4b shows the decomposition of the pre-waiting time into its flexible and fixed parts. Suppose

$$latest(v) < begin(v,u).$$

Then, the flexible part is the time interval $(early(v), latest(v))$ and the fixed part is the interval $(latest(v), begin(v,u))$. Although arc (v,u) is ready for execution no later than $latest(v)$, we have to wait until time $begin(v,u)$ because of the time-window constraint. Fig. 4c shows the decomposition of the post-waiting time into its flexible and fixed parts. If

$$early(u) > end(v,u) + t(v,u),$$

then the fixed part is the time interval $(end(v,u) + t(v,u), early(u))$ and the flexible part is the time interval $(early(u), latest(u))$. Although the activity of arc (v,u) can be finished at time $end(v,u) + t(v,u)$, we still have to wait until time $early(u)$ to begin the activities emanating from node u. This waiting time is fixed. The interpretation of the flexible part of the flow time is similar to that for time-schedule arcs.

Consider the time-window arc (B,E) in Fig. 2. We have $early(\mathrm{B}) = 5$, $latest(\mathrm{B}) = 6$, $t(\mathrm{B},\mathrm{E}) = 0.5$, $begin(\mathrm{B},\mathrm{E}) = 7$, $end(\mathrm{B},\mathrm{E}) = 8$, $early(\mathrm{E}) = 9$ and $latest(\mathrm{E}) = 10$. The calculations of the pre-waiting and post-waiting times and the decomposition of the flexible and fixed parts for this time-window arc are shown as follows:

The traditional flow time is

$$latest(\mathrm{E}) - early(\mathrm{B}) - t(\mathrm{B},\mathrm{E}) = 10 - 5 - 0.5 = 4.5.$$

The pre-waiting time is

$$begin(\mathrm{B},\mathrm{E}) - early(\mathrm{B}) = 7 - 5 = 2.$$

The post-waiting time is

$$latest(\mathrm{E}) - end(\mathrm{B},\mathrm{E}) - t(\mathrm{B},\mathrm{E}) = 10 - 8 - 0.5 = 1.5.$$

The float time is

$$latest(\mathrm{E}) - t(\mathrm{B},\mathrm{E}) - early(\mathrm{B}) - prew(\mathrm{B},\mathrm{E}) - postw(\mathrm{B},\mathrm{E}) = 1.$$

The fixed pre-waiting time is

$$begin(\mathrm{B},\mathrm{E}) - latest(\mathrm{B}) = 7 - 6 = 1.$$

The flexible pre-waiting time is

$$latest(\mathrm{E}) - early(\mathrm{E}) = 6 - 5 = 1.$$

The fixed post-waiting time is

$$early(\mathrm{E}) - end(\mathrm{B},\mathrm{E}) - t(\mathrm{B},\mathrm{E}) = 9 - 8 - 0.5 = 0.5.$$

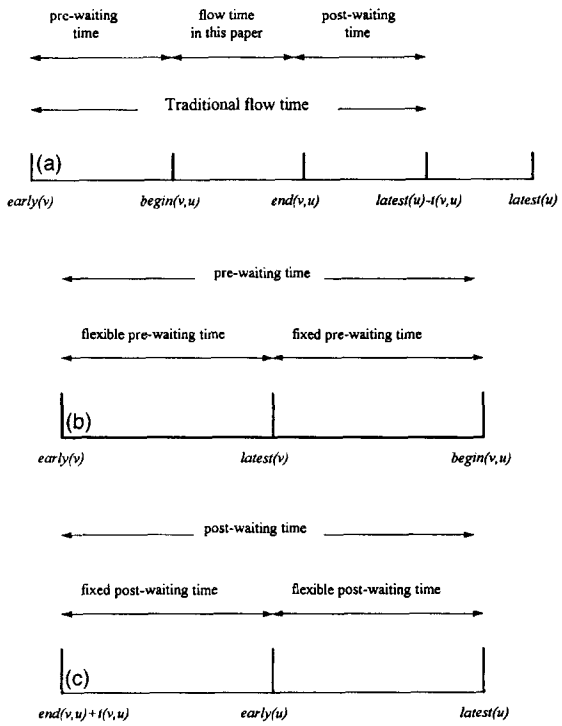

Fig. 4. (a) Different definitions of float time for normal arcs and time-window arcs; (b) decomposition of the pre-waiting time into two components; (c) decomposition of the post-waiting time into two components.

The flexible post-waiting time is

$$latest(\mathrm{E}) - early(\mathrm{E}) = 10 - 9 = 1.$$

4. Summary

In this paper, we incorporate two types of time constraints into the traditional activity network. The first is the time-window constraint, which requires that an activity be performed only in a specified time interval. The second is the time-schedule constraint, which requires that an activity can begin only at one of specified times. Since these two types of time constraints are often encountered in practice, the new model is more practical than the traditional model. The main results of this paper are summarized as follows.

1. *An efficient solution method is developed.* The solution method developed in this paper is a modification of the forward pass, backward pass algorithm used for traditional activity networks. Both phases of our algorithm can be run in time of $O(|A|+|V|)$, where $|A|$ and $|V|$ are the numbers of arcs and nodes in the network, respectively. From the time complexity point of view, algorithms which run in time $O(|A|+|V|)$ are called *optimum time algorithms*; that is, we cannot find another algorithm which has a better time complexity.

2. *The effects of the time constraints on the solution are identified.* These effects, which are given in Section 2, are mainly on the float times and waiting times associated with the arcs on the critical path. They provide important managerial insights to managing a project with time constraint consideration.

3. *Four meaningful components in the waiting time are identified.* The waiting time associated with an arc has been decomposed into pre-waiting time and post-waiting time. The pre-waiting time is the longest period we need to wait before beginning the activity of the arc, and the post-waiting time is the longest period we need to wait after finishing the activity before beginning another activity. The pre-waiting time and post-waiting time are further decomposed into a fixed part and a flexible part. The fixed part of a waiting time is unavoidable, whereas the project manager has scheduling discretion within the bounds of the flexible part. These four components have different managerial implications to project managers.

Appendix A

Theorem 1. *Let $early(v)$ denote the earliest time that all of the preceding activities on which node v depends are finished. Then the algorithm for the first phase correctly finds $early(v)$ for each node v in the node set V.*

Proof. We prove this theorem by induction. Let v_1 be the first node examined in the algorithm. Obviously, $v_1 = s$ and the theorem is true for node s since node s does not follow any other nodes. Assume the theorem is true for nodes $v_1, v_2, \ldots, v_{i-1}$. Now we consider node v_i whose topological order value is i. For node v_i, the value of $early(v_i)$ can be defined as

max{the time arc (u, v_i) reaches node $v_i \mid (u, v_i) \in A\}$. If arc (u, v_i) is normal, then the time of reaching v_i is $early(u) + t(u, v_i)$, because node u has a topological order value smaller than v_i and, by the induction assumption, $early(u)$ is already known. If arc (u, v_i) is time-schedule, then the reaching time is $ts_k(u, v_i) + t(u, v_i)$, where $ts_k(u, v_i)$ is the smallest scheduled time of arc $(u, v_i) \geq early(u)$, because, by the induction assumption, $early(u)$ is already known and $ts_k(u, v_i)$ is the earliest time for arc (u, v_i) to begin. Finally, if arc (u, v_i) is time-window, then there are three possible cases.

Case 1. If $early(u) < begin(u, v_i)$, then arc (u, v_i) must wait until time $begin(u, v_i)$. Therefore, the reaching time is $begin(u, v_i) + t(u, v_i)$.

Case 2. If $begin(u, v_i) \leq early(u) \leq end(u, v_i)$, then arc (u, v_i) can begin right away. Therefore, the reaching time is $early(u) + t(u, v_i)$.

Case 3. If $early(u) > end(u, v_i)$, then the problem is infeasible.

In Step 1.2 of the algorithm, we examine each arc (u, v_i) going into node v_i; use the variable $reaching(u, v_i)$ to store the reaching time of arc (u, v_i); compare this time with the currently known maximum value of $early(v_i)$ to see if the new value is larger; and update the value of $early(v_i)$ if required. Therefore, $early(v_i)$ denotes the earliest time that all of the activities preceding node v_i. are finished. □

Appendix B

Lemma 1. *For time-schedule arc (v_i, u), $ts_1(v_i, u)$ cannot be larger than $temp^{\&}(v_i, u)$ unless the problem is infeasible.*

Proof. If $ts_1(v_i, u) > temp^{\&}(v_i, u)$, then even if we begin the activity of arc (v_i, u) at time $ts_1(v_i, u)$, we cannot arrive at node u at time $latest(u)$. Because $latest(u) \geq early(u)$ and $ts_1(v_i, u)$ is the earliest possible time to begin arc (v_i, u), the earliest time to reach node u will be later than $early(u)$. Based on this contradiction, the lemma is proved. □

Lemma 2. *For time-window arc (v_i, u), $begin(v_i, u)$ cannot be larger than $temp^{\&}(v_i, u)$ unless the problem is infeasible.*

Proof. If $begin(v_i, u) > temp^{\&}(v_i, u)$, then even if we begin the work of arc (v_i, u) at time $begin(v_i, u)$, we cannot arrive at node u at time $latest(u)$. Because $latest(u) \geq early(u)$ and $begin(v_i, u)$ is the earliest possible time to begin arc (v_i, u), the earliest time to reach node u will be later than $early(u)$. This contradiction suggests that the lemma is true. □

Lemma 3. *Assume that the number of possible scheduled beginning times for each arc is a constant. Then, the time complexity of the algorithm is $O(|A| + |V|)$, where $|A|$ and $|V|$ are the numbers of arcs and nodes in the network, respectively.*

Proof. The procedure to find the topological order of all nodes requires time $O(|A| + |V|)$. In the first phase, we examine every node, and for each node we examine its incident arcs. Therefore, every node and every arc is examined just one time, and the time complexity for the first phase is $O(|A| + |V|)$. Similarly, the second phase has the same time complexity. Hence, the whole algorithm requires the time $O(|A| + |V|)$. □

References

[1] Adlakha, V.G., and Kulkarni, V.G. (1989), "A classified bibliography of research on stochastic PERT networks: 1966–1987", *INFOR* 27, 272–296.

[2] Anklesaria, K.P., and Drezner, Z. (1986), "A multivariate approach to estimating the completion time for PERT networks", *Journal of the Operational Research Society* 37, 811–815.

[3] Chae, K.C., and Kim, S. (1990), "Estimating the mean and variance of PERT activity time using likelihood-ratio of the mode and the midpoint", *IIE Transactions* 22, 198–203.

[4] Chen, Y.L., and Tang, K. (1994), "Shortest paths in time-schedule networks", Working Paper, Department of Information Systems and Decision Sciences, Louisiana State University.

[5] Cormen, T.H., Leiserson, C.E., and Rivest, R.L. (1992), *Introduction to Algorithms*, MIT Press/McGraw-Hill, New York, 485–488.

[6] Dumas, Y., Desrosiers, J., and Soumis, F. (1991), "The pickup and delivery problem with time windows", *European Journal of Operational Research* 54, 7–22.

[7] Hershaur, J.C., and Nabielsky, G. (1972), "Estimating activity times", *Journal of Systems Management* 23, 17–21.

[8] Kamburowski, J. (1985), "An upper bound on the expected

completion time of PERT networks", *European Journal of Operational Research* 21, 206–212.

[9] Keefer, D.L., and Verdini, W.A. (1993), "Better estimation of PERT activity time parameters", *Management Science* 39, 1086–1091.

[10] Magott, J., and Skudlarski, K. (1993), "Estimating the mean completion time of PERT networks with exponentially distributed durations of activities", *European Journal of Operational Research* 71, 70–79.

[11] Nadas, A. (1979), "Probabilistic PERT", *IBM Journal of Research and Development* 23, 339–347.

[12] Robillard, P., and Trahan, M. (1977), "The completion time of PERT networks", *Operations Research* 25, 15–29.

[13] Sculli, D. (1983), "The completion time of PERT networks", *Journal of the Operational Research Society* 34, 155–158.

[14] Solomon, M.M. (1987), "Algorithms for the vehicle routing and scheduling problems with time window constraints", *Operations Research* 35, 254–265.

[15] Soroush, H. (1993), "Risk taking in stochastic PERT networks", *European Journal of Operational Research* 67, 221–241.

[16] Soroush, H.M., (1994), "The most critical path in a PERT network", *Journal of the Operational Research Society* 45, 287–300.

[17] Williams, T.M. (1992), "Criticality in stochastic networks", *Journal of the Operational Research Society* 43, 353–357.