ELSEVIER

European Journal of Operational Research 120 (2000) 603–613

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/orms

Theory and Methodology

Finding the critical path in an activity network with time-switch constraints

Hsu-Hao Yang [a,*], Yen-Liang Chen [b]

[a] Department of Industrial Engineering and Management, National Chin-Yi Institute of Technology, Taiping, Taichung, Taiwan, ROC

[b] Department of Information Management, National Central University, Chungli, Taiwan, ROC

Received 1 September 1997; accepted 1 September 1998

Abstract

An activity network is an acyclic graph with non-negative weights and with a unique source and destination. A project consisting of a set of activities and precedence relationships can be represented by an activity network and the mathematical analysis of the network provides useful information for managing the project. In a traditional activity network, it is assumed that an activity always begins after all of its preceding activities have been completed. This assumption may not be adequate enough to describe some practical applications where some forms of time constraints are attached to an activity. In this paper, we investigate one type of time constraint called *time-switch constraint* which assumes that an activity begins only in a specified time interval of a cycle with some pairs of exclusive components. Polynomial time algorithms are developed to find the critical path (or longest path) and analyze the float of each arc in this time-constrained activity network. The analysis shows that the critical path and float in this context differ from those of a traditional activity network in some management perspectives and thus, consideration of the time-switch constraint leads to enhanced project management through more effective use of budgets and resource allocation. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Activity network; Critical path; Longest path; Time-constrained network

1. Introduction

An acyclic graph with non-negative weights and with a unique source and destination is called an activity network. In this network, an arc can represent an activity, while the precedence relationships between all of the activities are represented by the topology of the network. An arc leaving from a node cannot begin until all of the arcs going into this node have been completed. Given that a common objective is to find the longest path (or critical path), a delay of one activity on the path can delay the entire project's completion time. Using this information, the project manager can monitor critical activities more closely to ensure that the project meets the planned schedule. Another subject of concern is to analyze the float

* Corresponding author. Tel.: +886 4 3924505; fax: 886 4 3934620; e-mail: yanghh@chinyi.ncit.edu.tw

PII: S 0 3 7 7 - 2 2 1 7 (9 8) 0 0 3 9 0 - 7

(or slack) of each arc. Arcs on the longest path have zero float, while non-critical arcs have non-zero floats. The float of a non-critical arc indicates the degree of flexibility in completing the activity without affecting the project's completion. A non-critical arc with more float means that the execution of the arc is more adjustable. Therefore, the project manager can transfer resources from the non-critical activities to those more critical ones whenever necessary. Since float gives a measure of the flexibility in scheduling the activities without delaying the project's completion, it has an impact on two issues of concern: resource allocation and activity scheduling. Information pertaining to float is important to the project manager who is responsible for managing budgets and allocating resources to keep the progress of a project on schedule. For details and surveys, see [1].

Analysis of an activity network is generally concerned with scheduling issues, i.e., various time factor aspects. For instance, some of the research is concerned with estimating activity time more accurately [2–4], while another area of research deals with the stochastic nature of activity time [5–8]. There has been some controversy, however, about which path is the most critical one in a stochastic activity network [8,9]. Suffice to say, it is no easy task to correctly estimate the completion time of a project whenever stochastic activity times are considered [5,10–13]. Study of the time factor of an activity network extends to consider alternative outcomes in key nodes [14].

Though many aspects of the time factor have been studied, an important factor has not received much attention. In a traditional activity network, an arc is assumed to begin at any time, depending only on all of its preceding arcs being completed. In practice, however, some kinds of time constraint are usually associated with an activity and, hence, an activity is rarely ready for execution at any time. Failure to take time constraints into account may result in misleading information for the project management. Consider if we identify a critical path correctly, but ignore all the time constraints. Several problems may arise. First, the actual time of this path is likely to be much longer, because some critical arcs are dependent upon time constraints and can not be executed at any time. By the same reasoning, this path may not be the critical path because some other paths could have longer times when time constraints are included. Much worse, the solution may be infeasible because of time constraints. Finally, even if the solution remains unaffected when the time constraints are included, the float associated with each arc may be seriously misleading [15]. All the discussion suggests that we may manage a project in an inappropriate way if time constraints are ignored.

To more adequately deal with the problem, a recent paper by [15] considers two improvements over the traditional activity network by including two types of time constraints. The first type is the time-window constraint [16], which assumes that an activity to be executed only in a specified time interval. The second one is the time-schedule constraint [15], which requires that an activity can only be executed at one of an ordered schedule of the beginning times. An important finding by [15] is the interpretation of float when time constraints are present in an activity network. For instance, the arcs on the critical path may have positive float. By analyzing the nature of waiting time associated with an arc, the model provides more managerial insights than those in a traditional activity network for managing a project.

Based on the analysis and achievement of [15], in this paper we assume that time can be treated as repeating cycles where each cycle consists of two categories: (1) some pairs of *rest* and *work* windows; and (2) a *leading* number specifying maximal number of times each pair should iterate. In this context, activities can be executed in a work window, while activities can not be executed in a rest window. We name this kind of time constraint in an activity network as *time-switch constraint*. Situations with this kind of time constraint in practice are commonly encountered. For example, the typical schedule of a regular working day (Monday to Friday) is: 9–12 a.m. and 1–5 p.m., where a break takes place between 12 and 1 p.m. In a traditional activity network, the longest path denotes the most critical part of the project and should not be delayed. This property may not hold in a time-switch activity network as revealed by [15], which implies that we may have the option of executing the activities in some perspective other than simply monitoring critical activities. One such possible option is to execute the activities according to their priority. For example, handling customer complaints may not be a critical activity in completing a service but has a great impact on impressing the customers. Moreover, under the circumstances that the project's cost is directly measured by the total hours spent, we may reduce the cost if we start the activity somehow later. This can be easily observed from the example that starting at the break incurs unnecessary cost.

The remainder of this paper is as follows. In Section 2, the problem is formally defined and solution methods of two-phase (forward pass and backward pass) for a time-switch activity network are presented. Section 3 discusses the float in more details and explores some management perspectives when time constraints are considered. Conclusions and future extensions are in Section 4.

2. The problem and the solution method

Let $N = (V, A, t, s, d)$ be an activity-on-arc (AOA) network, where $G = (V, A)$ is a directed acyclic graph without multiple arcs, $t(u, v) \geqslant 0$ denotes the processing time of arc (u, v), s is the source node and d the destination node. A project represented by the current activity network is finished when each arc is completed. Arcs in this activity network can be classified into two categories: normal arcs and time-switch arcs. Let k and c be positive integers. A normal arc begins at any time after all of its preceding activities are completed. A time-switch arc is subject to the same constraint as a normal arc but with an additional constraint $\mathrm{TS}(u, v) = [c, (r_1, w_1), \ldots, (r_k, w_k)]$, where r_i (or w_i), $1 \leqslant i \leqslant k$, is the ith rest (or work) window in a cycle, c is the leading number, and k is the number of pairs. In this paper, we assume that each cycle always starts with a rest window; r_i (or w_i) may or may not equal r_j (or w_j) for $i \neq j$; and no preemption or interruption is allowed for a work window. The assumption that each cycle always starts with a rest window is founded on the premises that: (1) time is modeled as repeating cycles consisting of pairs of windows, each cycle starting with either a work window or a rest window; (2) the algorithms presented below can be easily modified for situations where the cycle starts with a work window (to be illustrated later). Therefore, this assumption does not restrict the applicability of the algorithms we propose. Because a normal arc is only a special type of a time-switch arc, all arcs are assumed to be with time-switch constraints in this paper. Take the regular working day mentioned in Section 1 as an example, then executing an activity can be represented as [1, (16, 3), (1, 4)] if the rest window before 9 a.m. starts from 5 p.m. of the previous day.

Fig. 1 shows an activity network with 7 nodes and 10 arcs, where all arcs are time-switch arcs. Activity times are shown on the arc, and the number inside each node is its topological order. Topological order is an order of nodes in an acyclic graph and can be easily constructed in time $O(|A| + |V|)$ by using the algorithm in [17], where $|A|$ and $|V|$ are the numbers of arcs and nodes in the network. If the topological order of node u is smaller than that of node v, all arcs emanating from node v cannot be predecessors to arcs emanating from node u. This ensures that all precedence relationships are satisfied when all the activities are executed according to their topological orders. To help understand examples follow, Fig. 2 shows rest windows (plain areas) and work windows (shaded areas) of each arc ranging from 0 to 36 based on the data given in Fig. 1.

Let $\mathrm{EA}(u)$ denote the earliest time that all predecessors to node u are completed in a time-switch activity network. If $\mathrm{EA}(u)$ falls inside a work window, arc (u, v) begins just at the time $\mathrm{EA}(u)$ without a delay. On the other hand, if $\mathrm{EA}(u)$ happens to be in a rest window, arc (u, v) cannot begin until the next work window. Owing to this, we need to develop a procedure to determine the earliest beginning time of arc (u, v). Let $\mathrm{EB}(u, v)$ be the earliest beginning time of arc (u, v), then the earliest finishing time of arc (u, v), denoted by $\mathrm{EF}(u, v)$, may not equal to $\mathrm{EB}(u, v) + t(u, v)$, because the activity may need more than one work window to complete. Consequently, we also need to develop a procedure to determine the earliest finishing time of arc (u, v).

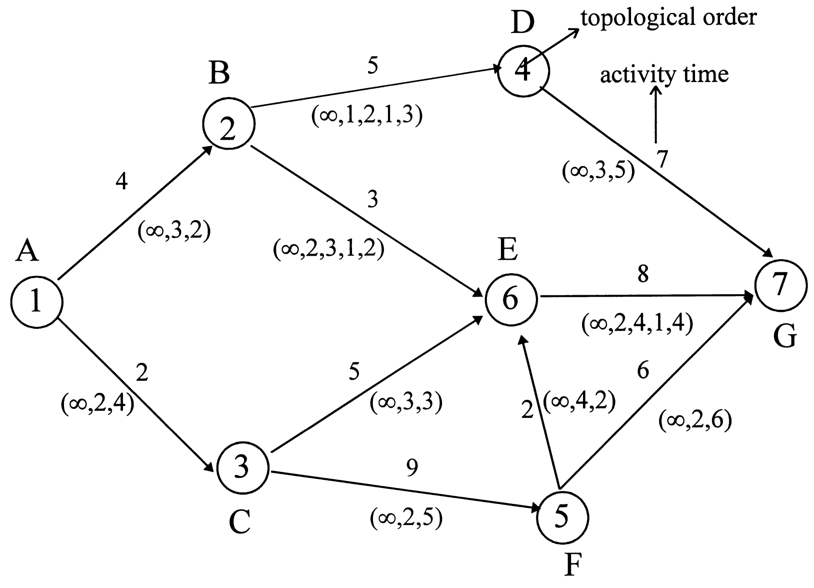

Fig. 1. An example of a time-switch network.

We prove a theorem (Appendix A, Theorem A.1) which validates the following algorithm for determining $\mathrm{EB}(u,v)$ and $\mathrm{EF}(u,v)$. In addition, the time complexity of the algorithm is shown to be O(k) in Appendix B, Lemma B.1. To present the algorithm, we introduce variables used in the algorithm first.

S_k	the total duration of a cycle with k pairs of rest and work windows, i.e., $$S_b = \sum_{i=1}^{b}(r_i + w_i), 1 \leqslant b \leqslant c \times k$$
WS_k	the work duration of a cycle with k pairs of rest and work windows, i.e., $$\mathrm{WS}_b = \sum_{i=1}^{b} w_i, 1 \leqslant b \leqslant c \times k$$
n	the number of complete k-pairs that EA(u) goes through
rem	the remaining time after EA(u) goes through n of k-pairs
j	rem falls inside either r_j or w_j
nw	the number of complete k-pairs required to finish arc (u,v)
rw	in Step 3.1 below, it means that after nw of work, how much more time is required to finish arc (u, v); in Steps 3.2 and 3.3, it denotes the remaining time required to finish arc (u,v)
r_{b+1}	the rest window follows w_b which is executing arc (u,v)
w_{b+1}	the work window follows r_{b+1}

Algorithm Earliest-Time.

```
1. (Initialization)
    Let $S_0 = 0$ and $\mathrm{WS}_0 = 0$.
    Compute $S_i = r_i + w_i + S_{i-1}$ and $\mathrm{WS}_i = w_i + \mathrm{WS}_{i-1}$ for $i = 1$ to $k$.
    Define $S_{(n\times k)+i} = (S_k \times n) + S_i$, $r_{(n\times k)+i} = r_i$, and $w_{(n\times k)+i} = w_i$ for all positive integers $n$ and for $i = 1$ to $k$.
2. (Find the earliest beginning time of arc $(u,v)$)
    2.1. Compute $n = \lfloor \mathrm{EA}(u)/S_k \rfloor$ and rem $= \mathrm{EA}(u)$ mod $S_k$.
    2.2. Find a $j$ such that $S_{j-1} \leqslant \mathrm{rem} < S_j$ where $1 \leqslant j \leqslant k$.
    As a result, $S_{(n\times k)+j-1} \leqslant \mathrm{EA}(u) < S_{(n\times k)+j}$.
    2.3. If $\mathrm{EA}(u) < S_{(n\times k)+j-1} + r_{(n\times k)+j}$, then $\mathrm{EB}(u,v) = S_{(n\times k)+j-1} + r_{(n\times k)+j}$
    else $\mathrm{EB}(u,v) = \mathrm{EA}(u)$.
```

```
3. (Find the earliest finishing time of arc $(u,v)$)
    3.1. Compute $nw = \lfloor t(u,v)/\mathrm{WS}_k \rfloor$ and $rw = t(u,v) \bmod \mathrm{WS}_k$.
    3.2. If $rw = 0$ and $\mathrm{EB}(u,v) = S_{(n\times k)+j-1} + r_{(n\times k)+j}$,
        then $\mathrm{EF}(u,v) = \mathrm{EB}(u,v) - r_{(n\times k)+j}$, go to Step 3.4
        If $rw < S_{(n\times k)+j} - \mathrm{EB}(u,v)$, then $\mathrm{EF}(u,v) = \mathrm{EB}(u,v) + rw$, go to Step 3.4
        $rw = rw - (S_{(n\times k)+j} - EB(u,v))$.
    3.3. For $b = n \times k + j$ to $(n+1) \times k + j - 1$
        if $w_{b+1} \geqslant rw$, then
            if $rw = 0$, then $\mathrm{EF}(u,v) = S_b$, go to Step 3.4
            else $\mathrm{EF}(u,v) = S_b + r_{b+1} + rw$, go to Step 3.4
        else $rw = rw - wb + 1$
        Endfor.
    3.4. $\mathrm{EF}(u,v) = \mathrm{EF}(u,v) + nw \times S_k$.
```

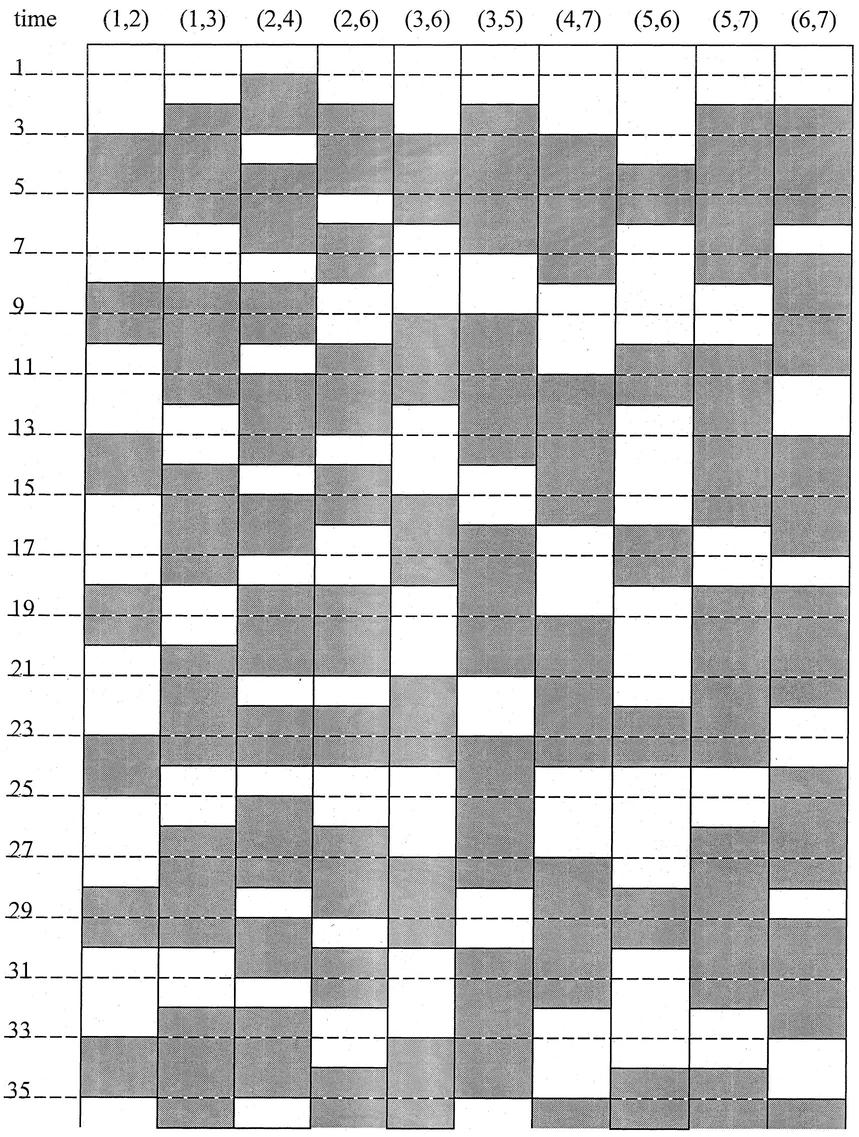

Fig. 2. The rest and work windows of all arcs in Fig. 1.

4. If $\mathrm{EF}(u,v) > c \times S_k$ then the solution is infeasible.

If the cycle starts with a work window, simply redefine S_i to be $w_i + ri + S_{i-1}$, and modify computations relevant to the window sequence. For example, Step 2.3 becomes: If $\mathrm{EA}(u) < S_{(n\times k)+j-1} + w_{(n\times k)+j}$, then $\mathrm{EB}(u,v) = \mathrm{EA}(u)$, else $\mathrm{EB}(u,v) = S_{(n\times k)+j}$. Refer to Theorem A. 1 where modifications are needed.

Example 1. Consider the arc (6, 7) in Fig. 1, where $t(6, 7) = 8$ and $\mathrm{TS}(6, 7) = [\infty, (2, 4), (1, 4)]$, i.e., $c = \infty$, $k = 2$, $r_1 = 2$, $w_1 = 4$, $r_2 = 1$, $w_2 = 4$. Suppose that $\mathrm{EA}(6) = 23$. By Step 1, $S_0 = 0$, $S_1 = 6$, $S_2 = 11$, $\mathrm{WS}_0 = 0$, $\mathrm{WS}_1 = 4$ and $\mathrm{WS}_2 = 8$. Step 2.1 computes $n = \lfloor 23/11 \rfloor = 2$, and rem = 23 mod 11 = 1. In Step 2.2, since $0 = S_0 \leqslant 1 = \text{rem} < S_1 = 6$, we have $22 = S_4 \leqslant 23 = \mathrm{EA}(6) < S_5 = 28$. Consequently, EA(6) falls inside either r_5 or w_5. Step 2.3 tests whether EA(6) is in r_5 by comparing EA(6) with $S_4 + r_5$ $(= 22 + 2 = 24)$. Since it is true (EA(6) < 24), EB(6, 7) is set to $S_4 + r_5 = 24$.

To compute EF(6, 7), by Step 3.1, $nw = \lfloor 8/8 \rfloor = 1$, and $rw = 8 \bmod 8 = 0$. Step 3.2 first checks whether rw can be finished in w_5. Since $rw = 0$, which means requiring none of w_5, $\mathrm{EF}(6,7) = \mathrm{EB}(6,7) - r_5 = 24 - 2 = 22$. Finally, by $nw = 1$ and $S_2 = 11$, Step 3.4 computes $\mathrm{EF}(6,7) = 22 + 1 \times 11 = 33$.

To analyze the float of the longest path in a time-switch network, we develop an algorithm similar to the two-phase procedure used in the traditional activity networks. The first phase (or forward pass) is to find the earliest beginning time of each node v, $\mathrm{EA}(v)$, and the longest path from s to d. On the other hand, the second phase (or backward pass) is to find the latest beginning time of every node v, denoted by $\mathrm{LL}(v)$, and analyze each arc's float. In the first phase, we examine every node v according to its topological order in ascending sequence, and use $P(v)$ to denote the preceding node to node v. Theorem A. 2 in Appendix A validates the following algorithm for finding $\mathrm{EA}(v)$. We also show the time complexity of the algorithm to be $O(|V| + |E|k)$ in Appendix B, Lemma A. 2.

Algorithm for the first phase.

```
1. Set $\mathrm{EA}(v_i) = 0$ for all nodes.
2. Examine every node $v_i$ in ascending sequence of topological order.
   2.1. For every arc $(u, v_i)$ going into node $v_i$, do
        if $\mathrm{EA}(v_i) < \mathrm{EF}(u, v_i)$, then
           $\mathrm{EA}(v_i) = \mathrm{EF}(u, v_i)$
           $P(v_i) = u$.
   2.2. For every arc $(v_i, u)$ leaving from node $v_i$, call Algorithm Earliest-Time to calculate $\mathrm{EB}(v_i, u)$ and $\mathrm{EF}(v_i, u)$.
3. Find the longest path by following the predecessor path from $d$ to $s$.
```

Example 2. Using the Algorithm for the first phase to solve the activity network in Fig. 1, the applying sequence of nodes is 1, 2, 3, 4, 5, 6, 7 and the final result is shown in Fig. 3, where the longest path is indicated by bold lines. If we examine node 6, we have $\mathrm{EF}(2,6) = 13$, $\mathrm{EF}(3,6) = 12$, $\mathrm{EF}(5,6) = 23$, and $\mathrm{EA}(6) = 0$. By Step 2.1, $\mathrm{EA}(6) = 23$, because $23 = \max\{13, 12, 23\}$. Since there is only one arc $(6,7)$ leaving from node 6, we only have to determine $\mathrm{EB}(6,7)$ and $\mathrm{EF}(6,7)$ by calling Algorithm Earliest-Time. Thus, we have $\mathrm{EB}(6,7) = 24$ and $\mathrm{EF}(6,7) = 33$. Remaining nodes are handled in the same way and omitted.

After presenting the algorithm for the first phase, we continue to discuss the second phase. Let *finish* be the duration of the longest path (in fact, *finish* = EA(d)). In the second phase, we examine every node v according to its topological order in descending sequence. Recall that $\mathrm{LL}(v)$ is the latest beginning time to leave node v while completing all remaining activities by the time *finish*. Suppose nodes $v_{|V|}, v_{|V|-1}, \ldots, v_{i+1}$ have been examined and now we examine node v_i, where the subscript denotes the topological order of a node. The value of $\mathrm{LL}(v_i)$ can be defined as

$$\underset{u}{\mathbf{Min}}\,\{\mathrm{LB}(v_i, u) | (v_i, u) \in A\},$$

where $\mathrm{LB}(v_i, u)$ denotes the latest time arc (v_i, u) must begin to complete all remaining activities by the time *finish*.

Suppose $\mathrm{LL}(v)$ is given, we are to find the latest time to finish arc (u, v) by going backward. If $\mathrm{LL}(v)$ falls inside a work window, the latest time to

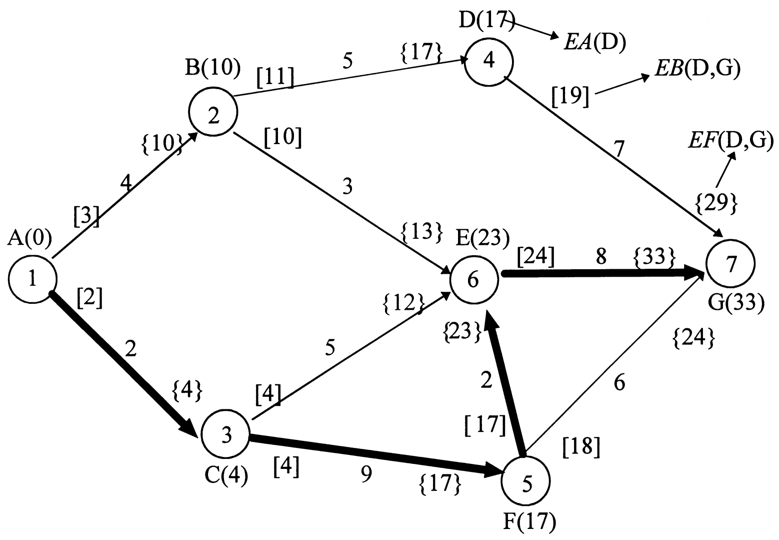

Fig. 3. The results of the forward pass of the algorithm.

finish arc (u,v) is LL(v), because later than this time contradicts to the definition of LL(v). On the other hand, if LL(v) falls inside a rest window, the latest time to finish arc (u,v) may be at the end of the previous work window. Hence, a procedure is required to determine the latest finish time of arc (u,v). Let LF(u,v) denote the latest finishing time of arc (u,v), then the latest beginning time of activity (u,v) may not equal to LF$(u,v) - t(u,v)$ as explained in the earlier. Therefore, we also need a procedure to determine the latest beginning time of arc (u,v). The following algorithm determines LF(u,v) and LB(u,v). The proof and time complexity of the algorithm are referred to those of Algorithm Earliest-Time.

Algorithm Latest-Time.

```
1. Same as **Algorithm Earliest-Time.**
2. (Find the latest finishing time of arc $(u, v)$)
    2.1. to 2.2. Same as **Algorithm Earliest-Time.**
    2.3. If $\mathrm{LL}(v) < S_{(n\times k)+j-1} + r_{(n\times k)+j}$, then
        $\mathrm{LF}(u,v) = S_{(n\times k)+j-1}$ else $\mathrm{LF}(u,v) = \mathrm{LL}(v)$.
3. (Find the latest beginning time of arc $(u,v)$)
    3.1. Same as **Algorithm Earliest-Time.**
    3.2. If $rw = 0$ and $\mathrm{LF}(u,v) = S_{(n\times k)+j-1}$,
        then $\mathrm{LB}(u,v) = \mathrm{LF}(u,v) + r_{(n\times k)+j}$, go to Step 3.4
        If $rw < \mathrm{LF}(u,v) - (S_{(n\times k)+j-1} + r_{(n\times k)+j})$, then $\mathrm{LB}(u,v) = \mathrm{LF}(u,v) - rw$ go to Step 3.4
        $rw = rw - [\mathrm{LF}(u,v) - (S_{(n\times k)+j-1} + r_{(n\times k)+j})]$.
    3.3. For $b = (n\times k) + j$ to $(n-1)\times k + j - 1$
        if $w_{b-1} \leqslant rw$, then
            if $rw = 0$, then $\mathrm{LB}(u, v) = S_b$, go to Step 3.4
            else $\mathrm{LB}(u, v) = S_b - 1 - rw$, go to Step 3.4
        else $rw = rw - w_{b-1}$
        Endfor.
    3.4. $\mathrm{LB}(u, v) = \mathrm{LB}(u, v) - nw \times S_k$.
```

Now, we discuss the algorithm for the second phase to compute the latest leaving time for every node, and the latest beginning and finishing times of every arc. The proof and time complexity of the algorithm are also referred to those of the Algorithm for the first phase.

Algorithm for the Second Phase.

```
1. Set $\mathrm{LL}(v_i) = \infty$ for all nodes. Set $\mathrm{LL}(d) = \mathrm{EA}(d)$, where $d$ is the destination node.
2. Examine every node $v_i$ in descending sequence of topological orders.
    2.1. For every arc $(v_i, u)$ leaving from node $v_i$, do
```

if $\mathrm{LL}(v_i) > \mathrm{LB}(v_i, u)$, then $\mathrm{LL}(v_i) = \mathrm{LB}(v_i, u)$.

2.2. For every arc (u, v_i) going into node v_i, call Algorithm Latest-Time to calculate $\mathrm{LB}(u, v_i)$ and $\mathrm{LF}(u, v_i)$.

Example 3. Based on the results of Fig. 3, we use Algorithm Latest-Time and the Algorithm for the second phase to solve the activity network in Fig. 1. Fig. 4 shows the final results, where the longest path is indicated by bold lines.

3. Discussions of floats

Since information about float is critical for a project manager who monitors the project to stay on schedule by adjusting budgets and allocating resources, in this section we provide a more detailed analysis of the float. Recall that the total float of an arc (u, v) in a traditional activity network is defined as

$$\mathrm{LL}(v) - \mathrm{EA}(u) - t(u, v).$$

To have an insight into the float and study how it may affect the scheduling of the project when time-switch constraints are included, we decompose the total float of an arc (u, v) into four parts: pre-waiting float, post-waiting float, in-waiting float and net float. Their definitions and calculations are given as follows:

- *pre-waiting float*: the time we are forced to wait before begin the activity ($= \mathrm{EB}(u, v) - \mathrm{EA}(u)$),
- *post-waiting float*: the time we are forced to wait after finish the activity ($= \mathrm{LL}(v) - \mathrm{LF}(u, v)$),
- *in-waiting float*: the time we are forced to wait during the execution of the activity ($= \mathrm{LF}(u, v) - \mathrm{LB}(u, v) - t(u, v)$),
- *net float*: the beginning time of the activity can vary ($= \mathrm{LB}(u, v) - \mathrm{EB}(u, v)$).

It is easy to see that the following equation holds:

$$\text{total float} = \text{pre-waiting float} + \text{in-waiting float} + \text{net float} + \text{post-waiting float}.$$

Consider the arc (4, 7) in Figs. 3 and 4, we have:

$$\mathrm{EA}(4) = 17, \quad \mathrm{EB}(4,7) = 19, \quad \mathrm{LB}(4,7) = 22,$$

$$\mathrm{EF}(4,7) = 29, \quad \mathrm{LF}(4,7) = 32, \quad \mathrm{LL}(7) = 33.$$

Though node 4 is ready for execution at time 17, it can not be executed until the next work window, time 19. Therefore, the pre-waiting float $= 19 - 17 = 2$. In an analogous way, the postwaiting float $= 33 - 32 = 1$. Suppose the activity begins at

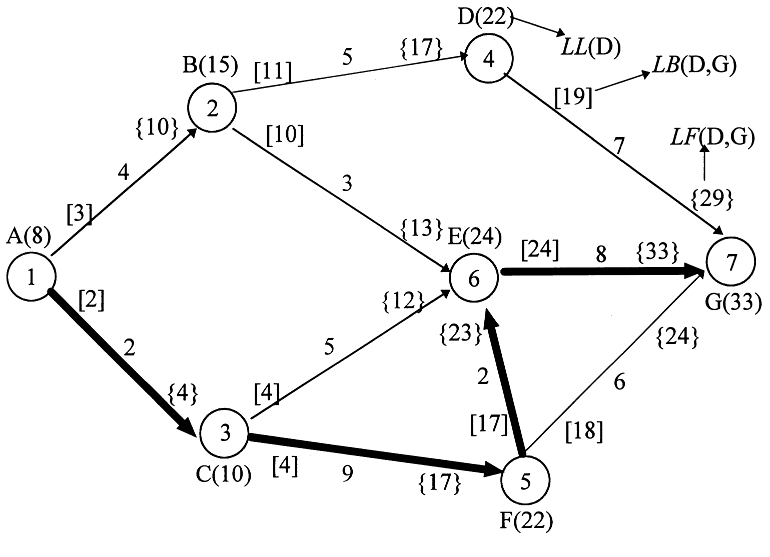

Fig. 4. The results of the backward pass of the algorithm.

time 22, it will finish at time 32 and thus the duration is 10. However, the actual working time is only $t(u,v) = 7$ and leaves the in-waiting float to be 3. Finally, the net float is $22 - 19 = 3$, which indicates that we have the flexibility of 3 to begin the activity.

After introducing four kinds of floats in the time-switch activity network, we present some interesting findings and discuss their implications different from those in a traditional activity network.

(1) *The beginning time of an arc may no longer be a single range of continuous values but contain multiple ranges of values*. Consider the arc (3, 6). We have

$$\mathrm{EB}(3,6) = 4, \quad \mathrm{LB}(3,6) = 16.$$

According to the definition, the net float equals 12. In a traditional sense, the activity can begin in range [4, 16] without delaying the entire project. In this example, however, the allowable beginning times are in ranges [4, 6), [9, 12), and [15, 16). On the one hand, the result simply reflects the nature of the problem. On the other hand, it supplements information for effectively allocating the resources by properly meeting the separate schedule.

(2) *The net float of an arc may not remain a constant*. Consider an alternate form of definitions of the net float and in-waiting float:

- *in-waiting float*: the time we are forced to wait during the execution of the activity ($= \mathrm{EF}(u,v) - \mathrm{EB}(u,v) - t(u,v)$),
- *net float*: the finishing time of the activity can vary ($= \mathrm{LF}(u,v) - \mathrm{EF}(u,v)$),

where the total float remains equal to the sum of four types of float. Interestingly, the net floats in these two viewpoints could be different. Consider the arc (3, 5):

$$\mathrm{EB}(3,5) = 4, \quad \mathrm{LB}(3,5) = 10,$$
$$\mathrm{EF}(3,5) = 17, \quad \mathrm{LF}(3,5) = 21.$$

According to the original definition the net float is $10 - 4 = 6$, in contrast, the alternate form shows that the net float equals $21 - 17 = 4$. What leads to these two values unequal is because different beginning times require different work and rest windows. In this example, beginning at time 4 will finish the activity at time 17, while 4 out of 13 are in the rest window. In contrast, beginning at time 10 will finish the activity at time 21, leaving only 2 in the rest window. The result suggests that beginning earlier may require *actual* working times longer, an interesting outcome compared with a traditional model where the length of actual working time is generally insensitive to the beginning time. From a management perspective where the project's cost is related to the actual working time, the cost can be controlled more precisely using this information.

(3) *The total float of a critical arc may no longer be zero*. It is obvious from (2) the arc (3, 5) is on the longest path but with a positive float. What does it imply that a critical arc may have a positive float? From the perspective of scheduling, this finding suggests that a critical arc is likely to be delayable, i.e., to execute a critical activity but with relaxed time constraints. From the perspective of resource allocation, it indicates that an arc on the longest path is less critical than that in a traditional one. Whether or not to pay the most attention to the critical arcs should actually depend on the progress or needs of the project. That is, we may execute the activities based on a priority list in order of their floats.

(4) *The whole system may have a system's float, because the latest beginning time of the entire project might be greater than zero*. For example, Fig. 4 has the following:

$$\mathrm{EA}(1) = 0, \quad \mathrm{EA}(7) = 33,$$
$$\mathrm{LB}(1,3) = 8, \quad \mathrm{LL}(7) = 33.$$

Given the data, if the project begins at the earliest time 0, then it takes 33 time units to finish the project. However, if the project does not begin until time 8, it still can finish at time 33, only requiring 25 time units. The finding is in principal consistent with that of (2) above, since a project is composed of a set of activities with time-switch constraints. The implication is that we may have an option of evaluating the entire project's cost from an overall perspective without knowing each activity's detailed float.

4. Conclusions

Managing a project with effective allocation of budgets and resources is important in practical applications. Since time factors play a dominant role in scheduling and managing the project, failure to include them not only produces misleading information but also results in incorrect decisions. In this paper, we incorporates one type of time constraint, called time-switch constraint, into the traditional network. The main results of this paper are summarized as follows.

(1) *A more practical and useful model is proposed.* In a traditional activity network, an activity is executed at any time after all of its preceding activities have been completed. In reality, however, an activity is rarely ready for execution at any time because of some forms of time constraints. Inclusion of such time constraints is more valuable and practical.

(2) *Efficient solution methods have been developed.* Both the forward pass and the backward pass algorithms for determining the critical activities are developed and can be run in time of $O(|A| + k|V|)$. Because of the algorithms' efficiency, they can be applied without causing serious computational difficulties.

(3) *The differences from a management perspective between the traditional model and the proposed model are investigated.* By decomposing the traditional total float into four components, the investigation reveals the following findings: (a) the critical arcs may have floats, (b) the entire project may also have the float, (c) the float of an arc may consist of multiple ranges of values, and (d) the net float of an arc can be viewed in two ways.

This paper has some possible extensions. First, the window time can be assumed to be uncertain in nature. Moreover, to reflect the importance of controlling the project's cost, a setup cost associated with executing an activity can be considered. Finally, as discussed in the earlier section that the system may have a system's float, two interesting issues deserve further studying: (1) what is the shortest critical path in a time-switch network? and (2) when to begin the project with minimum actual time to finish the project?

Acknowledgements

The authors are grateful to the anonymous referees for their helpful comments.

Appendix A

Theorem A.1. *Let* EB(u, v) *and* EF(u, v) *denote the earliest beginning time and finishing time of arc* (u, v). *Then the algorithm Earliest-Time correctly finds* EB(u, v) *and* EF(u, v) *for each arc* (u, v) *in the arc set* A.

Proof. The central logic of the algorithm consists of two parts: (1) determines the number of k-pairs (S_k) advanced, and (2) finds the remaining time and the window. To compute EB(u, v), Steps 2.1 and 2.2 obtain n, rem, and j as described in (1) and (2). Then, there are two cases.

Case 1. If EA(u) is in a work window, then arc (u,v) can begin right away; therefore, $\mathrm{EB}(u,v) = \mathrm{EA}(u)$.

Case 2. If EA(u) is in a rest window, then arc (u,v) has to delay until the next work window; therefore, $\mathrm{EB}(u,v) = S_{(n\times k)+j-1} + r_{(n\times k)+j}$.

To compute $\mathrm{EF}(u,v)$, Step 3.1 does (1) and (2) and obtains nw and rw. After obtaining nw and rw, we first find the finishing time based on rw and $\mathrm{EB}(u,v)$, and then add this finishing time to the result of multiplying S_k by nw. There are three cases to be considered.

Case 1. If $rw = 0$ and $\mathrm{EB}(u,v) = S_{(n\times k)+j-1} + r_{(n\times k)+j}$, then $S_{(n\times k)+j-1}$ is the end of work; therefore, $\mathrm{EF}(u,v) = \mathrm{EB}(u,v) - r_{(n\times k)+j}$.

Case 2. If $rw < w_b$ (i.e., current work window), then arc (u,v) can be finished in this window; therefore, $\mathrm{EF}(u,v) = \mathrm{EB}(u,v) + rw$.

Case 3. If $rw > w_b$, then we must iteratively search subsequent $k - 1$ work windows, each by reducing rw, to finish arc (u,v). In each iteration, there are two cases.

- If $w_{b+1} > rw$ and $= 0$, then w_b is the end of $\mathrm{EF}(u,v)$; therefore, $\mathrm{EF}(u,v) = S_b$.
- If $w_{b+1} > rw$, then $\mathrm{EF}(u,v)$ can be finished in w_{b+1}; therefore, $\mathrm{EF}(u,v) = S_b + r_{b+1} + rw$.

Finally, it is obvious the solution is infeasible if $\mathrm{EF}(u,v) > c \times S_k$. □

Theorem A.2. *Let* EA(v) *denote the earliest time that all predecessors to node v are completed. Then the algorithm for the first phase correctly finds* EA(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 because node s is the source node. Assume the theorem is true for nodes $v_1, v_{2\dots}, v_{i-1}$. Now we consider node v_i whose topological order is i. For node v_i, the value of EA(v_i) can be defined as $\mathbf{Max}_u$ {the earliest time arc (u, v_i) reaches node $v_i | (u, v_i) \in A\}$. Since EF(u, v_i) found by Algorithm Earliest-Time is the earliest time arc (u, v_i) reaches node v_i, the definition of EA(v_i) can be rewritten as $\mathbf{Max}_u$ $\{\mathrm{EF}(u, v_i) | (u, v_i) \in A\}$. Since Step 2.1 matches this definition, EA(v_i) is truly the earliest time that all of the activities preceding node v_i are completed. □

Appendix B

Lemma B.1. *Algorithm Earliest-Time has time complexity* O(k), *where k is the maximum number of pairs of windows in a cycle.*

Proof. Steps 1 and 2.2 can be done in time O(k), while Steps 2.1, 2.3, 3.1, 3.2, 3.4 and 4 can be done in time O(1). Because the loop in Step 3.3 iterates from $b = n \times k + j$ to $b = (n + 1) \times k + j - 1$, the algorithm can be done in time O(k). □

Lemma B.2. *The time required for the first phase is* $\mathrm{O}(|V| + |E|k)$, *where k is the maximum number of pairs of windows in a cycle.*

Proof. The first phase examines every node one time and every arc two times. Further, we call Algorithm Earliest-Time when initially examining each arc. □

References

[1] S.E. Elmaghraby, Activity nets: A guided tour through some recent developments, European Journal of Operational Research 82 (3) (1995) 383–408.

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

[3] J.C. Hershaur, G. Nabielsky, Estimating activity times, Journal of Systems Management 23 (9) (1972) 17–21.

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

[5] J. Magott, K. Skudlarski, Estimating the mean completion time of PERT networks with exponentially distributed durations of activities, European Journal of Operational Research 71 (1) (1993) 70–79.

[6] A. Nadas, Probabilistic PERT, IBM Journal of Research and Development 23 (3) (1979) 339–347.

[7] H. Soroush, Risk taking in stochastic PERT networks, European Journal of Operational Research 67 (2) (1993) 221–241.

[8] T.M. Williams, Criticality in stochastic networks, Journal of the Operational Research Society 43 (4) (1992) 353–357.

[9] H.M. Soroush, The most critical path in a PERT network, Journal of the Operational Research Society 45 (3) (1994) 287–300.

[10] P.K. Anklesaria, Z. Drezner, A multivariate approach to estimating the completion time for PERT networks, Journal of the Operational Research Society 37 (8) (1986) 811–815.

[11] J. Kamburowski, An upper bound on the expected completion time of PERT networks, European Journal of Operational Research 21 (2) (1985) 206–212.

[12] P. Robillard, M. Trahan, The completion time of PERT networks, Operations Research 25 (2) (1977) 15–29.

[13] D. Sculli, The completion time of PERT networks, Journal of the Operational Research Society 34 (2) (1983) 155–158.

[14] D. Golenko-Ginzburg, D. Blokh, A generalized activity network model, Journal of the Operational Research Society 48 (4) (1997) 391–400.

[15] Y.L. Chen, D. Rinks, K. Tang, Critical path in an activity network with time constraints, European Journal of Operational Research 100 (1997) 122–133.

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

[17] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction To Algorithms, MIT Press, Cambridge, MA, 1992, pp. 485–488.