
Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

# Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks

Prabuddha De, E. James Dunne, Jay B. Ghosh, Charles E. Wells,

To cite this article:
Prabuddha De, E. James Dunne, Jay B. Ghosh, Charles E. Wells, (1997) Complexity of the Discrete Time-Cost Tradeoff Problem for Project Networks. Operations Research 45(2):302-306. https://doi.org/10.1287/opre.45.2.302

Published in Operations Research on April 01, 1997 as DOI: 10.1287/opre.45.2.302.
This article has not been copyedited or formatted. The final version may differ from this version.





# COMPLEXITY OF THE DISCRETE TIME-COST TRADEOFF PROBLEM FOR PROJECT NETWORKS

**PRABUDDHA DE, E. JAMES DUNNE, JAY B. GHOSH and CHARLES E. WELLS**

*University of Dayton, Dayton, Ohio*

(Received August 1992; revisions received July 1993, April 1994; accepted October 1994)

This note addresses the discrete version of the well-known time-cost tradeoff problem for project networks, which has been studied previously in the standard project management literature as well as in the related literature on Decision-CPM. All the algorithms proposed thus far for the solution of the general problem exhibit exponential worst-case complexity, with the notable exception of the pseudo-polynomial dynamic program due to Hindelang and Muth. We first demonstrate that this algorithm is flawed, and that when we correct it, it no longer remains pseudo-polynomial. Continuing on in the main result of the note, we show that this is not at all surprising, since the problem is strongly NP-hard. Finally, we discuss the complexities of various network structures and validate an old conjecture that certain structures are necessarily more difficult to solve.

Consider a project network of $n$ activities in activity-on-node representation. Let two dummy activities correspond to the start and the finish of the project, and number the nodes 0 through $n + 1$ such that a node's number is smaller than each of its successors. Assume that an activity $i$, $i = 1, \ldots, n$, has $m(i)$ alternatives of which alternative $j$, $j = 1, \ldots, m(i)$, requires $t(i, j)$ time and $c(i, j)$ cost. Further assume, without loss of generality, that $t(\cdot, \cdot)$ and $c(\cdot, \cdot)$ are integers, and that if $k$ and $r$ are alternatives for activity $i$ such that $k < r$, then $t(i, k) < t(i, r)$ and $c(i, k) > c(i, r)$. Figure 1 shows a simple project network.

Let $\sigma$, $\sigma \equiv \{\langle i, j\rangle, i = 1, \ldots, n\}$, denote a specific realization of the project where alternative $j$ is chosen for activity $i$ and $ct(\sigma)$ and $cc(\sigma)$ denote, respectively, the early finish time and the cost of that realization. Let $\Theta$ be the set of the $\Pi_{1 \leq i \leq n}\, m(i)$ possible realizations, $b$ be a budget, and $d$ be a due date for the project. The basic objective is to identify a realization $\sigma^*$ from $\Theta$ which minimizes $ct(\cdot)$ subject to $cc(\cdot) \leq b$ or one which minimizes $cc(\cdot)$ subject to $ct(\cdot) \leq d$. Typically, however, the above problems are solved parametrically for all $b \in [\underline{b}, \bar{b}]$ ($d \in [\underline{d}, \bar{d}]$)—$\underline{b}$ ($\underline{d}$) and $\bar{b}$ ($\bar{d}$) being, respectively, the lowest (shortest) and the highest (longest) cost (time) in which the project can be realized—to obtain the project time-cost curve. This is what we call the *discrete time-cost tradeoff* (DTCT) problem in this note.

DTCT has been studied in the traditional time-cost tradeoff context; see Meyer and Shaffer (1965), Butcher (1967), Robinson (1975), Panagiotakopoulos (1977), Harvey and Patterson (1979), Bein et al. (1992), Elmaghraby (1993), De et al. (1993), Tufekci (1993), and Demeulemeester et al. (1993), among others. It has also been addressed in the Decision-CPM context; see, for example, Crowston and Thompson (1967), Crowston (1970), and Hindelang and Muth (1979).

All the reported algorithms that correctly solve DTCT exhibit an exponential worst-case complexity. A notable exception is the dynamic program due to Hindelang and Muth, which executes in pseudo-polynomial time. Thus, if this algorithm were correct, it would in effect have solved the problems in polynomial time regardless of the network structure, as long as the time and cost parameters remained bounded by a polynomial in the problem size.

In this note we demonstrate that such is not the case. First, we show that the Hindelang-Muth algorithm is flawed. We also offer a simple correction, but fail to maintain the pseudo-polynomial time guarantee. This, however, is not surprising, since we prove next, as our main result, that DTCT is strongly NP-hard (Garey and Johnson 1979). We then examine its complexity for special network structures and validate an old conjecture (Butcher) that the problem is more difficult for certain networks (those containing the so-called forbidden graph; see Figure 1).

## 1. THE HINDELANG-MUTH ALGORITHM

For the reader's convenience, we briefly describe the Hindelang-Muth algorithm (we call it HM_DP) as adapted for regular project networks. Let $P(i)$ and $S(i)$ be, respectively, the sets of the immediate predecessors and immediate successors of node $i$. Further, let $\underline{s}(i)$ and

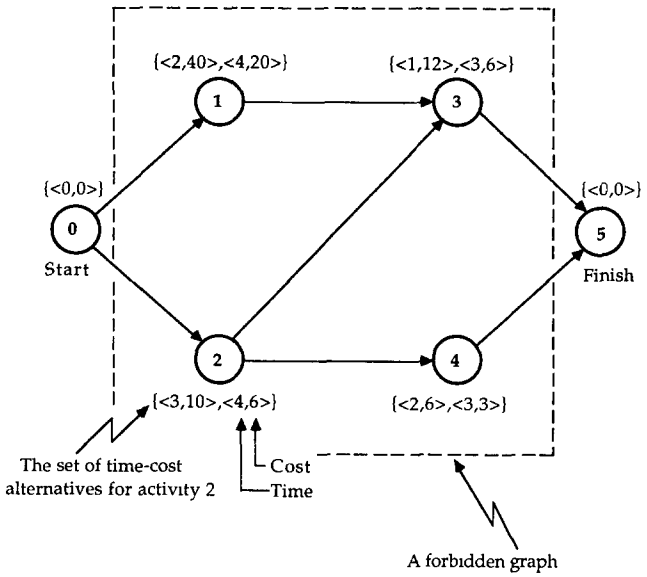
*Subject classifications:* Project management: time-cost tradeoff. Analysis of algorithms, computational complexity: strong NP-completeness.
*Area of review:* MANUFACTURING, OPERATIONS AND SCHEDULING.


0030-364X/97/4502-0302 $05.00

Published in Operations Research on April 01, 1997 as DOI: 10.1287/opre.45.2.302.
This article has not been copyedited or formatted. The final version may differ from this version.








Note: Project due-date, d = 6.

**Figure 1.** Example of a project network.

$\underline{s}(i)$ and $\bar{s}(i)$ be, respectively, the early and late start times for node $i$ relative to $\bar{d}$. Define $f(i, s)$ to be the minimum cost of realizing node $i$ and its successors such that node $i$ starts at time $s$ and all nodes complete by time $\bar{d}$; let $\delta(i, s)$ record the alternative that yields $f(i, s)$. The dynamic programming recursion, for $i = n, \ldots, 0$, is given by

$$f(i, s) = \min_{1 \leq j \leq m(i)} \left\{ \left[ \sum_{i' \in S(i)} f(i', s + t(i, j)) \right] + c(i, j) \right\}, \quad \text{for } \underline{s}(i) \leq s \leq \bar{s}(i),$$

$f(i, s) = f(i, \underline{s}(i))$ for $s < \underline{s}(i)$, and

$f(i, s) = \infty$ otherwise.

The initial conditions are: $f(n + 1, s) = 0$ for $s \leq \bar{s}(n + 1)$, and $f(n + 1, s) = \infty$ otherwise.

The HM_DP algorithm executes in three steps:

*Step 1.* $\underline{s}(i)$ and $\bar{s}(i)$, $i = 0, \ldots, n + 1$, are computed in $O(n^2)$ time.

*Step 2.* The dynamic programming recursion is carried out in $O(\bar{m}n^2\bar{d})$ time, where $\bar{m} = \max_{1 \leq i \leq n}\{m(i)\}$. Special care is taken to avoid the double counting of cost. If $i$ happens to be a merge node (i.e., $|P(i)| \geq 2$), then only one from among all the predecessors of $i$ should inherit $f(i, \cdot)$, the accumulated cost up to $i$. Thus, once $f(i', \cdot)$ is computed and $\delta(i', \cdot)$ recorded, $f(i', \cdot)$ is set to $c(i', \delta(i', \cdot))$ for each $i' \in P(i)$ except one.

*Step 3.* An optimal realization is obtained for each feasible start time of node 0. The set of these optimal realizations delivers a solution to DTCT. The complexity of this step is $O(n^2\bar{d})$.

The complexity of the HM_DP algorithm as a whole is seen to be $O(\bar{m}n^2\bar{d})$, which is pseudo-polynomial.

Now, consider the project network of Figure 1 and assume that the objective is to identify a minimum cost realization which completes by the due-date of 6. The HM_DP solution in this case, if node 2 is made to inherit the accumulated cost at node 3, is $\sigma \equiv \{\langle 1, 2\rangle, \langle 2, 1\rangle, \langle 3, 1\rangle, \langle 4, 2\rangle\}$ with $ct(\sigma) = 6$ and $cc(\sigma) = 45$. It can be verified, however, that the correct solution is given by $\sigma^* \equiv \{\langle 1, 2\rangle, \langle 2, 2\rangle, \langle 3, 1\rangle, \langle 4, 1\rangle\}$ with $ct(\sigma^*) = 6$ and $cc(\sigma^*) = 44$.

Let $i$ be a nontrivial merge node, viz., a merge node for which different start times are possible at different costs. HM_DP fails because it does not account for the fact that nodes $i'$, $i'' \in P(i)$ may anticipate distinctly different start times for node $i$. This difficulty can be avoided by using multiple passes of HM_DP such that every nontrivial merge node is frozen to a single start time in a given pass. In the network of Figure 1, node 3 is the only nontrivial merge node with $\underline{s}(3) = 3$ and $\bar{s}(3) = 5$; in this case, a correct solution can be guaranteed with three passes of HM_DP, one for each feasible start time of node 3.

The complexity of the multiple-pass correction to the HM_DP algorithm is $O(\bar{m}n^2\bar{d}^w)$, where $w$ is the number of nontrivial merge nodes. Since $w$ can be arbitrarily large and may approach $n$, the above complexity cannot be claimed as being pseudo-polynomial.

## 2. PSEUDO-POLYNOMIAL INSOLVABILITY

We now show that a pseudo-polynomial algorithm for DTCT is indeed unlikely, since the problem is strongly NP-hard. To this end, we introduce the DTCT decision problem: is there a $\sigma \in \Theta$ such that $ct(\sigma) \leq d$ and $cc(\sigma) \leq b$? Note that for now we refer to this problem as the DTCT problem and the original DTCT problem as the DTCT optimization problem.

We also introduce the exactly-one-in-three 3SAT problem, which we call EOIT_3SAT and which is strongly NP-complete (Garey and Johnson, p. 259): Given $p$ variables $\{V(j), j = 1, \ldots, p\}$ and $q$ clauses $\{C(i), i = 1, \ldots, q\}$ with each clause representing a disjunction of three literals (where a literal $U$ is either $V$ or $\bar{V}$), is there a truth assignment for the variables in $\{V(j), j = 1, \ldots, p\}$ such that each clause in $\{C(i), i = 1, \ldots, q\}$ has exactly one true literal?

From the above instance of EOIT_3SAT, we then construct an instance of DTCT, where the project network consists of $3p + 3q + 2$ nodes (nodes 0 and $3p + 3q + 1$ are the dummy start and finish nodes), $d$ is equal to 2, and $b$ equal to $Mp + 2q$ (we let $M$ equal $3q + 1$).

The first $3p$ real nodes of the network are called the variable nodes: we use a three-node cluster comprising nodes $3j - 2$, $3j - 1$ and $3j$ to model the truth assignment to variable $V(j)$ and its complement $\overline{V(j)}$, $j = 1, \ldots, p$. Figure 2(a) describes the structure of this part of the network. Notice that the alternatives for node $3j - 2$ are given by the set of time-cost pairs $\{\langle 0, M\rangle, \langle 1, 0\rangle\}$, those

Published in Operations Research on April 01, 1997 as DOI: 10.1287/opre.45.2.302.
This article has not been copyedited or formatted. The final version may differ from this version.



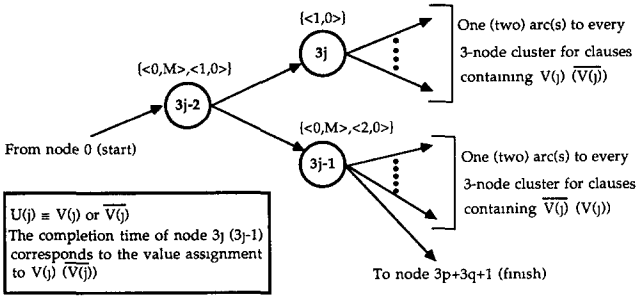



**Figure 2.** (a) The three-node cluster corresponding to variable $V(j)$.

for node $3j - 1$ are given by $\{\langle 0, M\rangle, \langle 2, 0\rangle\}$, and for node $3j$ by $\{\langle 1, 0\rangle\}$. The execution of nodes $3j - 2$ and $3j - 1$ control the truth assignment to $V(j)$ and $\overline{V(j)}$, respectively. We say that a particular assignment is TRUE if the associated node is executed using the lower-cost alternative. It is easy to verify that in order to simultaneously satisfy the requirements that $ct(\sigma) \leq 2$ and $cc(\sigma) \leq Mp + 2q$, it is necessary that exactly one of the nodes $3j - 2$ and $3j - 1$, $j = 1, \ldots, p$, is executed using the lower-cost alternative (in fact, the arc from node $3j - 1$ to node $3p + 3q + 1$ and the relatively high value of $M$ ensure this). Thus, DTCT has a solution only if the variable nodes are executed in a way that is consistent with a valid truth assignment to $V(j)$ and $\overline{V(j)}$, $j = 1, \ldots, p$. Finally, note that the truth assignment to $V(j)$ and $\overline{V(j)}$ is reflected in the completion times of nodes $3j$ and $3j - 1$, respectively: TRUE (FALSE) corresponds to a completion time of 1 (2) at the associated node. Nodes $3j$ and $3j - 1$ are actually the nodes that are connected to the second part of the network and determine the early start times of the nodes in that part.

The last $3q$ real nodes of the network are called the clause nodes: we use a three-node cluster comprising nodes $3p + 3i - 2$, $3p + 3i - 1$ and $3p + 3i$ to model a clause $C(i)$, $i = 1, \ldots, q$. Figure 2(b) describes the structure of this part of the network. Notice that each node in a cluster has the same set of alternatives given by $\{\langle 0, 1\rangle, \langle 1, 0\rangle\}$. Thus, if a node's early start time (which is the maximum of its predecessors' completion times) is 1, the node is executed using the lower-cost alternative; else, (i.e., if its early start time is 2), it has to be executed using the higher-cost alternative. Assume that clause $C(i)$ involves the literals $U(x)$, $U(y)$ and $U(z)$. For this clause, node $3p + 3i - 2$ will have as its predecessors the nodes associated with $\overline{U(x)}$, $\overline{U(y)}$ and $U(z)$ in the first part of the network. (For example, if $U(x) = V(x)$, $U(y) = \overline{V(y)}$ and $U(z) = V(z)$, then nodes $3x - 1$, $3y$ and $3z$ will be the predecessors of node $3p + 3i - 2$; see the construction for $C(1)$ in Figure 3.) Similarly, nodes $3p + 3i - 1$ and $3p + 3i$ will have as their predecessors the node sets associated with $\{\overline{U(x)}, U(y), \overline{U(z)}\}$ and $\{U(x), \overline{U(y)}, \overline{U(z)}\}$, respectively. Figure 3, alluded to earlier, shows the complete network for a three-variable, two-clause instance of EOIT_3SAT, and illustrates, in particular, the interconnections between the two parts of a network. Finally, Table I shows the early start times of the three nodes in the cluster for clause $C(i)$ corresponding to all the valid truth assignments to $U(x)$, $U(y)$ and $U(z)$ induced by the execution of the associated variable nodes; recall that these truth assignments are reflected in the completion times (either 1 or 2) of the node pairs $\langle 3x, 3x - 1\rangle$, $\langle 3y, 3y - 1\rangle$, and $\langle 3z, 3z - 1\rangle$ in the first part of the network.

Clearly, the above instance of DTCT is constructed in time that is polynomial in the length of the given instance of EOIT_3SAT, and has a length that is polynomially related to the length of that instance. It is also clear that the

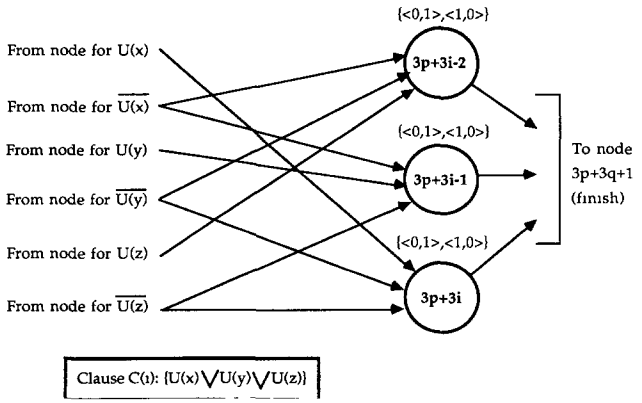



**Figure 2.** (b) The three-node cluster corresponding to clause $C(i)$.

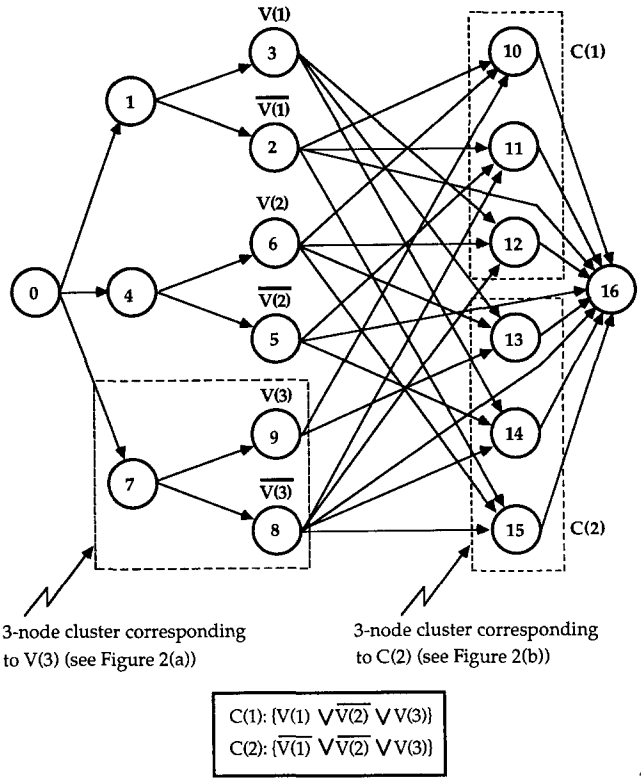



**Figure 3.** Network for a three-variable, two-clause instance of EOIT_3SAT.

Published in Operations Research on April 01, 1997 as DOI: 10.1287/opre.45.2.302.
This article has not been copyedited or formatted. The final version may differ from this version.



**Table I**
Early Start Times for the Three-node Cluster Corresponding to Clause $C(i)$ as a Function of Truth Assignments

| Truth Assignments | | | Early Start Times | | |
|---|---|---|---|---|---|
| $U(x)$ | $U(y)$ | $U(z)$ | Node $3p + 3i - 2$ | Node $3p + 3i - 1$ | Node $3p + 3i$ |
| True | True | True | max{2, 2, 1} = 2 | max{2, 1, 2} = 2 | max{1, 2, 2} = 2 |
| False | True | True | max{1, 2, 1} = 2 | max{1, 1, 2} = 2 | max{2, 2, 2} = 2 |
| True | False | True | max{2, 1, 1} = 2 | max{2, 2, 2} = 2 | max{1, 1, 2} = 2 |
| True | True | False | max{2, 2, 2} = 2 | max{2, 1, 1} = 2 | max{1, 2, 1} = 2 |
| False | False | True | max{1, 1, 1} = 1 | max{1, 2, 2} = 2 | max{2, 1, 2} = 2 |
| False | True | False | max{1, 2, 2} = 2 | max{1, 1, 1} = 1 | max{2, 2, 1} = 2 |
| True | False | False | max{2, 1, 2} = 2 | max{2, 2, 1} = 2 | max{1, 1, 1} = 1 |
| False | False | False | max{1, 1, 2} = 2 | max{1, 2, 1} = 2 | max{2, 1, 1} = 2 |

*Notes:* Literal $U \equiv V$ or $\bar{V}$, where $V$ is a variable.
Clause $C(i)$: $\{U(x) \vee U(y) \vee U(z)\}$.
$\{U$ = True$\} \equiv$ {Associated node completes at time 1}.
$\{U$ = False$\} \equiv$ {Associated node completes at time 2}.

largest number in the DTCT instance ($M$) is bounded by a polynomial function of the largest number in the EOIT_3SAT instance ($\max\{p, q\}$).

We now claim that the constructed instance of DTCT has a solution if and only if the given instance of EOIT_3SAT has a solution. First, we suppose that EOIT_3SAT has a solution, and use the truth assignment given by this solution to execute the variable nodes. The total cost due to the execution of these nodes clearly equals $Mp$. Further, Table I suggests that the early start time for exactly one among the three nodes in a three-node cluster for each clause is 1; only such a node is executed using the lower-cost alternative, yielding a total cost of $2q$ for the clause nodes. The realization that we have generated for the network is feasible, has a cost of $Mp + 2q$, and thus shows that the constructed instance of DTCT has a solution.

Next, we suppose that the DTCT instance has a solution. We have observed before that if this is the case, the execution of the variable nodes must correspond to a valid truth assignment. Thus, we assign truth to a variable based upon whether its associated node has been executed using the lower-cost or the higher-cost alternative. Clearly, in the DTCT solution, the total cost due to the variable nodes is $Mp$. Table I indicates that the total cost due to the clause nodes must therefore be exactly $2q$, which is possible only if the three-node cluster for each clause has a cost of 2. It is again clear from Table I that this is possible only if the truth assignment is such that each clause has exactly one true literal. The truth assignment generated by us is valid, leads to exactly one true literal in each clause, and thus shows that the given instance of EOIT_3SAT has a solution.

With the above, we have in effect provided a pseudo-polynomial transformation (Garey and Johnson, p. 101) from EOIT_3SAT to DTCT. We know that EOIT_3SAT is strongly NP-complete. In addition, DTCT clearly is in NP, since we can check in $O(n^2)$ time whether a given realization of the network provides a solution to DTCT. This completes our proof that DTCT is strongly NP-complete.

Strong NP-completeness of the DTCT decision problem implies that the DTCT optimization problem is strongly NP-hard. It cannot thus be solved by a pseudo-polynomial time algorithm unless it is true that $P = NP$ (Garey and Johnson, p. 95), which is unlikely.

## 3. CLOSING REMARKS

While DTCT in general cannot be solved effectively, its special cases can be. For a pure parallel network, DTCT is solvable in polynomial time, e.g., in $O(\bar{m}n)$ time by a merge and scan algorithm; in this case, HM_DP delivers a correct solution in $O(\bar{m}n\bar{d})$ time. For a pure series network, DTCT can be shown to be ordinary NP-hard through a transformation from the knapsack problem (Garey and Johnson, p. 247). DTCT in this case can, however, be solved in pseudo-polynomial time, e.g., $O(\bar{m}n\bar{d})$ time by the HM_DP algorithm.

Valdes et al. (1982) show that a series-parallel network can be recognized and decomposed hierarchically in $O(n)$ time into parallel and series modules. For a general network, Buer and Mohring (1983) and Muller and Spinrad (1989), among others, propose a modular decomposition which can similarly decompose the network in $O(n^2)$ time into parallel, series, and neighborhood (which is neither parallel nor series) modules. The fact that all nodes in a module share the same precedence relationship with all other nodes, together with the nature of decomposition, enables us to solve DTCT for an entire network by independently solving DTCT for the various modules in a bottom-up order. Since the number of compositions in a network is $O(n)$, DTCT for series-parallel networks can be solved in pseudo-polynomial time, e.g., in $O(\bar{m}n^2\bar{d})$ time with the HM_DP algorithm.

The strong NP-hardness of DTCT suggests at this point that for nonseries-parallel networks (those containing the neighborhood module), DTCT must necessarily be more difficult to solve. It follows immediately from Valdes et al. that these difficult networks are precisely the ones that also

have embedded forbidden graphs. This observation validates an old conjecture (see Butcher) about the difficulty of solving such networks.

The situation is not totally hopeless for nonseries-parallel networks. At least as long as the number of nontrivial merge nodes in a neighborhood module that constitutes such a network is bounded by a constant, the corrected HM_DP algorithm will provide a pseudo-polynomial solution; when this constant is small, the solution will also be practical. For further details on the solution of nonseries-parallel networks, refer to the papers by Bein et al., Elmaghraby, De et al., and Demeulemeester et al.

## REFERENCES

Bein, W. W., J. Kamburowski, and M. F. M. Stallmann. 1992. Optimal Reduction of Two-Terminal Directed Acyclic Graphs. *SIAM J. Comput.* **21,** 1112–1129.

Buer, H. and R. H. Mohring. 1983. A Fast Algorithm for the Decomposition of Graphs and Posets. *Math. O. R.* **8,** 170–184.

Butcher, W. S. 1967. Dynamic Programming for Project Cost-Time Curves. *J. Construction Div., Proc. ASCE*, **93,** 59–73.

Crowston, W. B. 1970. Decision CPM: Network Reduction and Solution. *O. R. Quarterly*, **21,** 435–452.

Crowston, W. B. and G. L. Thompson. 1967. Decision CPM: A Method for Simultaneous Planning, Scheduling, and Control of Projects. *Opns. Res.* **15,** 407–426.

De, P., E. J. Dunne, J. B. Ghosh, and C. E. Wells. 1993. The Discrete Time-Cost Tradeoff Problem Revisited. Working Paper, The Center for Business and Economic Research, University of Dayton, Dayton, Ohio.

Demeulemeester, E. L., W. S. Herroelen, and S. E. Elmaghraby. 1993. Optimal Procedures for the Discrete Time/Cost Trade-Off Problem in Project Networks. Working Paper, Department of Applied Economics, Katholieke Universiteit Leuven, Leuven, Belgium.

Elmaghraby, S. E. 1993. Resource Allocation via Dynamic Programming in Activity Networks. *European J. Opnl. Res.* **64,** 199–215.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability*. W. H. Freeman and Company, New York.

Harvey, R. T. and J. H. Patterson. 1979. An Implicit Enumeration Algorithm for the Time/Cost Tradeoff Problem in Project Network Analysis. *Foundations of Control Engineering*, **4,** 107–117.

Hindelang, T. J. and J. F. Muth. 1979. A Dynamic Programming Algorithm for Decision CPM Networks. *Opns. Res.* **27,** 225–241.

Meyer, W. L. and L. R. Shaffer. 1965. Extending CPM for Multiform Project Time-Cost Curves. *J. Construction Div., Proc. ASCE*, **91,** 45–65.

Muller, J. H. and J. Spinrad. 1989. Incremental Modular Decomposition. *J. ACM*, **36,** 1–19.

Panagiotakopoulos, D. 1977. A CPM Time-Cost Computational Algorithm for Arbitrary Activity Cost Functions. *INFOR*, **15,** 183–195.

Robinson, D. R. 1975. A Dynamic Programming Solution to Cost-Time Tradeoff for CPM. *Mgmt. Sci.* **22,** 158–166.

Tufekci, S. 1993. On the Complexity of Time Cost Tradeoff Problems. Working Paper, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL.

Valdes, J., R. E. Tarjan, and E. L. Lawler. 1982. The Recognition of Series-Parallel Digraphs. *SIAM J. Comput.* **11,** 298–313.