विद्याविनियोगाद्विकासः

IIM AHMEDABAD

Working Paper

N APPROXIMATE ALGORITHM FOR REDUCING DUMMY-ACTIVITIES IN A PERT NETWORK

By

OMPRAKASH K. GUPTA

W P No. 793

March, 1989

The main objective of the working paper series of the IIMA is to help faculty members to test out their research findings at the pre-publication stage.

INDIAN INSTITUTE OF MANAGEMENT
AHMEDABAD 380 056
INDIA

AN APPROXIMATE ALGORITHM FOR REDUCING DUMMY-ACTIVITIES IN A PERT NETWORK

Omprakash K. Gupta
Indian Institute of Management
Ahmedabad 380 056

ABSTRACT:

A project is an enterprise consisting of several activities which are to be carried out in some specific order. The activities and the order in which they need to be carried out can be represented by a pert network. Two types of networks are commonly used: Activity-On-Node (AON) and Activity-On-Arrow (AOA) networks. When networks are used, it often becomes necessary to draw dummy activities. Since the computation of project completion time is proportional to the number of arcs, including dummy, it is desirable to draw a network with as few dummy activities as possible. It has been earlier shown that the minimum-dummy-activities problem is NP-complete. In this paper we propose an approximate algorithm for solving the dummy activities problem. The algorithm is explained by an example.

AN APPROXIMATE ALGORITHM FOR REDUCING DUMMY-ACTIVITIES IN A PERT NETWORK

Omprakash K. Gupta
Indian Institute of Management
Ahmedabad 380 056

ABSTRACT:

A project is an enterprise consisting of several activities which are to be carried out in some specific order. The activities and the order in which they need to be carried out can be represented by a pert network. Two types of networks are commonly used: Activity-On-Node (AON) and Activity-On-Arrow (AOA) networks. When networks are used, it often becomes necessary to draw dummy activities. Since the computation of project completion time is proportional to the number of arcs, including dummy, it is desirable to draw a network with as few dummy activities as possible. It has been earlier shown that the minimum-dummy-activities problem is NP-complete. In this paper we propose an approximate algorithm for solving the dummy activities problem. The algorithm is explained by an example.

1. INTRODUCTION

A project is an enterprise consisting of several activities which are carried out in some specified order. The activities and the order in which they are required to be carried out may be represented by a network. Two types of networks are commonly used for representing projects: Activity-On-Arrow (AOA) networks and Activity-On-Node (AON) networks. The former are also called the activity networks and the latter the pert or event networks. In case of Activity-On-Node type of representation there is a unique network without redundant arcs. In case of Activity-On-Arrow networks, one has to often draw some dummy activities in order to satisfy the precedence relationships.

Therefore there may be numerous ways of drawing AOA networks of a given project. Since we are interested in finding the critical path and compute the project completion time, it is desirable to have network with as few dummy activities as possible as the computation time required to do the network analysis is proportional to the number of arcs (including dummy).

The problem of constructing an AOA network with minimum number of dummy activities has been studied [1,2,3,4,5,6] in Operations Research and Computer Science. Krishnamoorthy and Deo [4] have shown that the problem of the minimum number of dummy activities in an AOA network which realizes the given set of precedence relations is NP-complete. Therefore, they have suggested, it would be worthwhile to look for polynomial-time approximate algorithms instead of an exact algorithm. Syslo [6] has later remarked that an approximation procedure may produce some dummy activities even if they may not be necessary. He has further argued that it is worthwhile to examine whether a network needs any dummy activity at all! He has characterized precedence-relations where dummy activities are not required and proved this question can be answered in polynomial-time.

In real-life projects, however, it almost always becomes necessary to draw dummy activities. In this paper, we, therefore, address ourselves to the question of developing an approximate algorithm for drawing an AOA network with as few dummy activities as possible.

2. DEFINITIONS

Suppose that the project consists of activities $a_1, a_2, \ldots\ldots\ldots, a_n$ alongwith a set of irredundant precedence relationships among them. We wish to construct a pert network with minimum number of dummy activities. In such a network activities will be on edges (arcs), and the nodes will represent events (milestones). A given activity will therefore be uniquely represented as an ordered-pair of nodes. We will label the initial and terminal nodes of activity a (a_j) by positive integers I_a (I_j), and T_a (T_j), respectively. We will follow the following conventions:

(1) All node numbers will be distinct.

(2) $I_j < T_j$ for each activity a_j.

(3) If $a_j < a_k$ (i.e. a_j is an immediate predecessor of a_k), then $T_j < I_k$.

(4) The network will have one global initial node,. and one global terminal node.

(5) No two activities will have the same initial and terminal nodes.

We will partition the project activities in levels $L_1, L_2, \ldots\ldots, L_m$ as follows:

(i) $L_1 = \{a/a$ is a project activity and it has no predecessor$\}$

(ii) $L_j = \{a/a$ is a project activity. If $b < a$, then $b \in L_i$, where $i<j$ and there exists an activity $c<a$ such that $c \in L_{j-1}$. $[j>1]$

We would call a node a free-node if no activity has that node as its initial node. For example, in Figure 1, nodes 2 and 5 are free nodes.

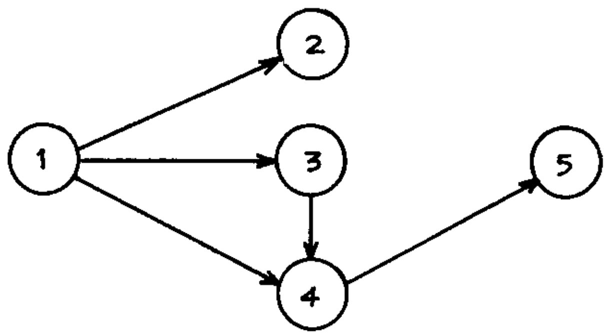


Fig. 1

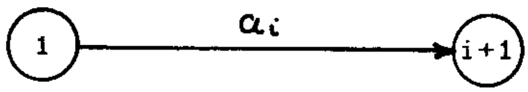
3. APPROXIMATE ALGORITHM

1. Partition the project activities into Levels. Let m be the number of the highest-level. Denote the number of activities in the Level L_i by N_i. Redesignate, if necessary, so that the activities $a_1, a_2, \ldots, a_{N_1}$ are in L_1; activities $a_{N_1+1}, a_{N_1+2}, \ldots, a_{N_2}$ are in L_2; and so on.

2. Create node representation of activities in Level L_1 as follows:

   If $a_i \in L_1$, then represent it by $I_i = 1$, $T_i = i+1$.

   Therefore the node-representation of the activity a_i would be:



   Let LAST denote the highest integer used for node-representation. Therefore, LAST = N_1+1.

3. If there are no more levels left, go to step 5. Otherwise, go to the next-higher level.

4. Partition all activities of this level in two groups. First group will have activities with exactly one predecessor, and the second group will have activities with two or more predecessors. Activities with exactly one predecessor will be node-represented first as follows:

Suppose b is an activity with one (and only) predecessor, say a. Since a is an activity in the preceeding level, it must have been given a node-representation with initial node I_a and terminal node T_a. We now represent b as: $I_b = T_a$, and T_b= LAST+1, and then update the value of LAST.

Activities with two or more predecessors can be represented in the following manner.

Suppose we wish to represent an activity b which has two or more predecessors. If an activity a is a predecessor of the activity b, it would have already been given a node-representation with I_a as the initial node and T_a as the terminal node. We next scan for a free node among the terminal nodes of predecessors of b. There are two possibilities:

A) At least one of the terminal nodes is a free node,

B) None of the terminal nodes is a free node.

CASE A:

Among the free terminal nodes, select the node with the largest node number, say LARGE. Represent the activity b by: I_b = LARGE, T_b = LAST+1. Update the value of LAST.

We next need to connect other terminal nodes to node I_b in order to satisfy precedence relationships. Consider one such terminal node, and suppose it has been given the number T. The node T is certainly the terminal node of at least one of the predecessors of b. It is however possible that it may also be the terminal node of some other activities. Next check if all the activities for which T is the terminal node are also predecessors of the activity b. If so, check whether T is a free node. If T is not a free node, create a dummy activity $(T) \cdots\cdots\cdots \rightarrow (I_b)$. If T is a free node, examine the initial nodes of all the activities for T is the terminal node. Let I be any such initial node. If the network does not contain any activity $(I) \longrightarrow (I_b)$, then T can be extended to I_b by replacing T by I_b. In case there exists an activity a for which T is the terminal node and I is the initial node and the network does contain an activity with initial node I and the terminal node I_b, create a dummy $(T) \cdots\cdots \rightarrow (I_b)$.

If there exists an activity for which T is the terminal node but it is not a predecessor of the activity b, we would not be in a position to create such a dummy activity from node T to node I_b. We however would have to represent I_b as a subsequent successor of T. This can be accomplished by following:

1) If a is a predecessor of b with T as its terminal node, change its node representation by replacing its terminal node T by a new node, say N. Thus a is now represented as:

2) Create dummy activities 

, and 

.

CASE B:

Suppose no terminal node of the predecessors of activity b is a free node. The activity b can be represented by: I_b = LAST+1, T_b = LAST+2, and update the value of LAST. We would now need to make node connections so that the predecessors of the activity b would have I_b as a subsequent successor node. Procedure as explained in Case A can be used for this purpose.

All the activities of this level can be represented using the above scheme.

Go to Step 3.

5. We have by now node-represented all the activities. There could, however, be several end-nodes. Since we wish to have only one global terminal node, we extend all the end-nodes to the largest-numbered node as follows:

Consider an end-node, say with number T. Therefore T is the terminal node of one or more activities. Consider an activity a for which T is the terminal node, and I_a is the initial node. For every a for which T is the terminal node, if the network does not contain any activity with representation (I_a) ⟶ (LAST), replace T by LAST. Otherwise, draw a dummy (T) ·········> (LAST).

6. Resequence the node numbers if necessary to ensure that if $a_j < a_k$, then $T_j < I_k$.

4. EXAMPLE

Consider the following example to illustrate the algorithm proposed in the above section.

Activity	Predecessors
A	–
B	–
C	–
D	A
E	A
F	A, B
G	B, C
H	F, G

First we partition these 8 activities into levels.

Level	Activities
1	A, B, C
2	D, E, F, G
3	F, G

Level 1:

Represent the activities A,B,C as: (1) —A→ (2), (1) —B→ (3), and (1) —C→ (4). The value of LAST would be 4. We would have the following network (Fig. 2):

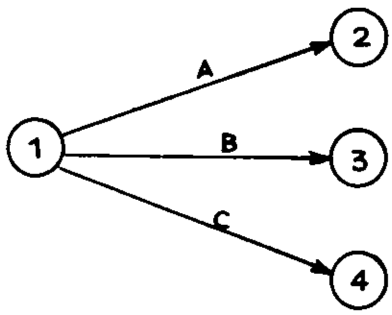
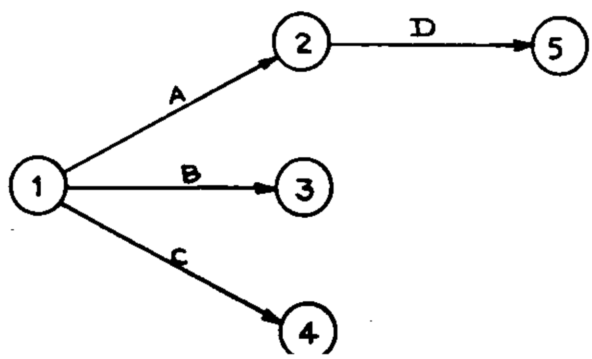
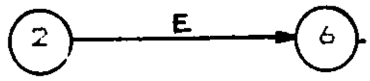


Fig. 2

Level 2:

This level has four activities D, E, F, and G. First consider activities D and E which have exactly one predecessor. Consider activity D which has activity A as its predecessor. Since activity A has been presented as (1) —A→ (2), represent D as: (2) —D→ (5). Update LAST as 5. We now have the following network (Fig. 3):



Next, take activity E. This activity also has one predecessor, namely activity A. Therefore, as done in the case of activity D, represent E as: 

.

Update LAST to 6 and network to following (Fig. 4):

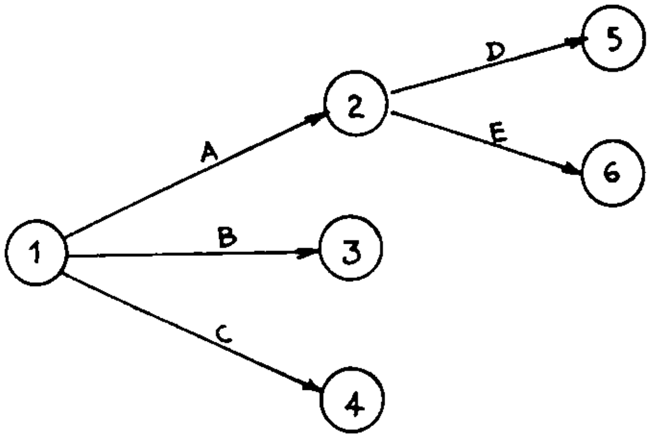
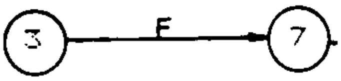
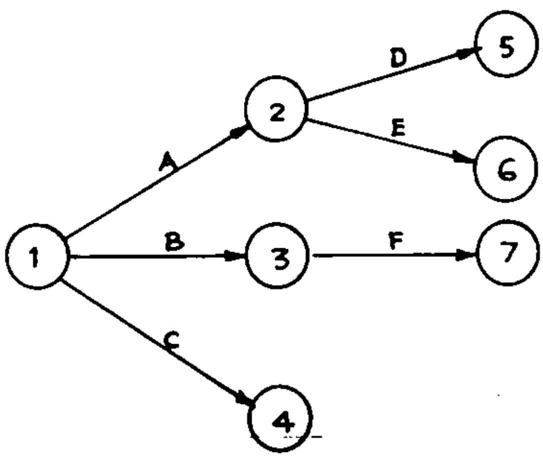


Fig. 4

Next consider activity F which has two predecessors, namely A and B. The terminal nodes of A and B are 2 and 3, respectvely. Since node 3 is a free node, represent F as:



. Update LAST to 7. The resulting network is shown in Figure 5.



Since A is also a predecessor of F, we must make node 2 precede node 3. There is only one activity, namely activity a, for which node 2 is the terminal node. The initial node of the activity a is node 1. We cannot, however represent A as: (1) ———→ (3) since we have already used this representation for the activity B. Therefore, create a dummy d_1 as: (2) .....d_1....→ (3) . (See Figure 6):

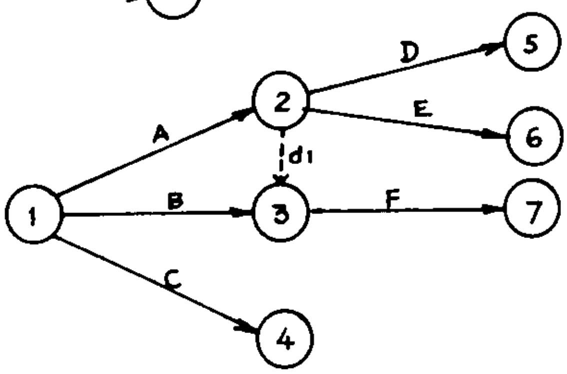
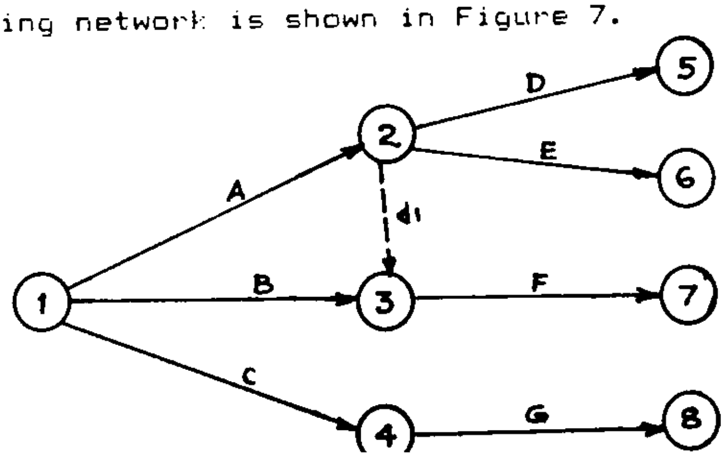


Fig. 6

Next consider the activity G for which the activities B and C are predecessors. The terminal nodes of B and C are node 3 and 4, respectively. Since node 4 is free node, we represent G as: (4) ——G——→ (8), and update LAST to 8. The resulting network is shown in Figure 7.



Since B is also a predecessor of G, we must make node 3 precede node 4. Node 3 has two activities for which it is the terminal node. Though the activity B is a predecessor of G, the dummy activity d_1 is not. We change the representation of B as: (1) —B→ (9) and create dummy activities d_2: (9) ....d_2...→ (3), and d_3: (9) .....d_3...→ (4). The resulting network is shown in Figure 8.

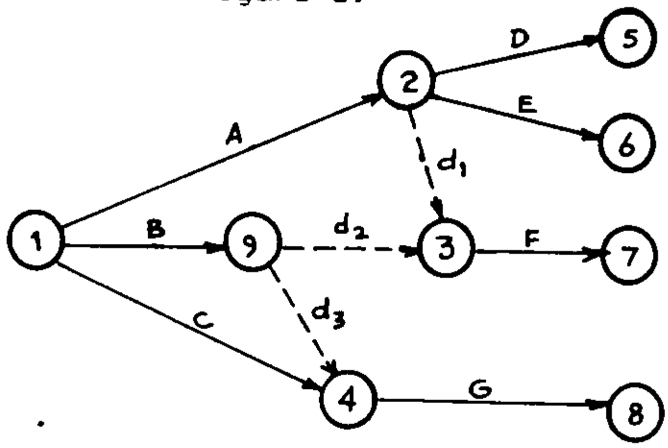


Fig. 8

Level 3:

This level has only one activity H with predecessors F and G. Since both of them are free nodes, select the highest numbered node (which is node 8), and represent H as: (8) —H→ (10). The resulting network is shown in Figure 9.

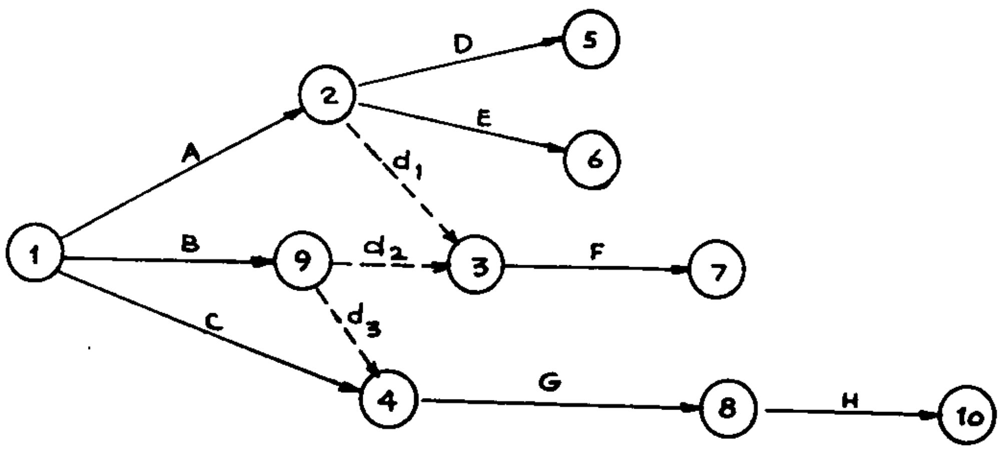


Fig. 9

Since F is also a predecessor of H , we need to examine node 7 which is the terminal node of F. Since F is the only activity for which node 7 is the terminal node, we can connect node 7 to node 8. Since node 3 is the initial node of F and there is no activity represented as: (3) —F→ (8) , we can simply extend node 7 to node 8 by replacing node 7 by node 8. The resulting network is shown in Figure 10.

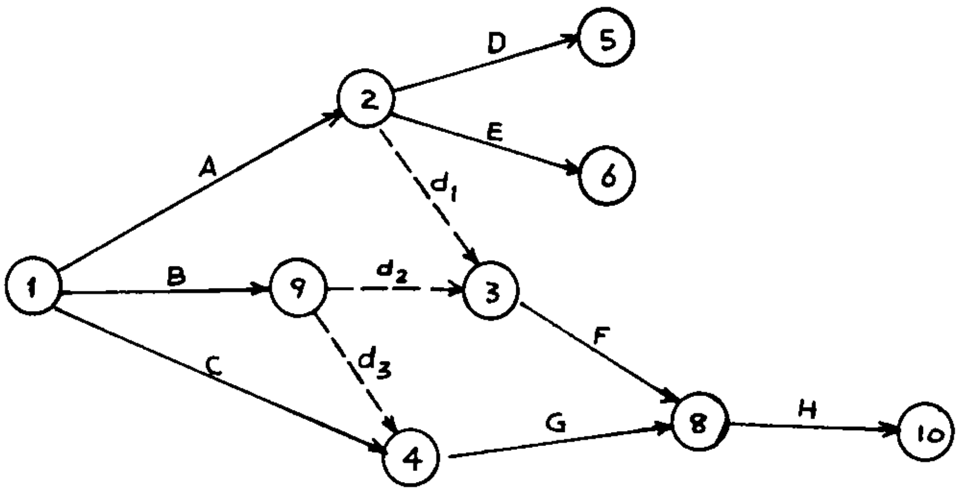


Fig. 10

At this stage we have represented all activities. We next consider extending the end-nodes 5, and 6 to node 10. If we extend node 5 to node 10, D will have representation: (2) —D→ (10) . Since no activity is currently represented as: (2) ——→ (10), we extend node 5 to node 10. Next, we consider extending node 6 to node 10. If we do so, the activity E will be represented by (2) ——→ (10). Since D has already been given this representation, we cannot extend 6 to 10. Therefore we create a dummy d_4: (6) ..d_4..→ (10). The resulting network is shown in Figure 11.

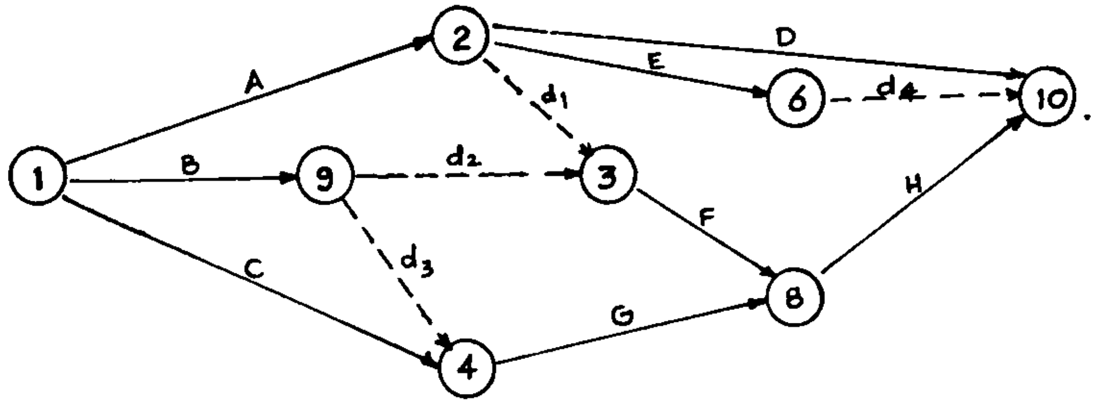
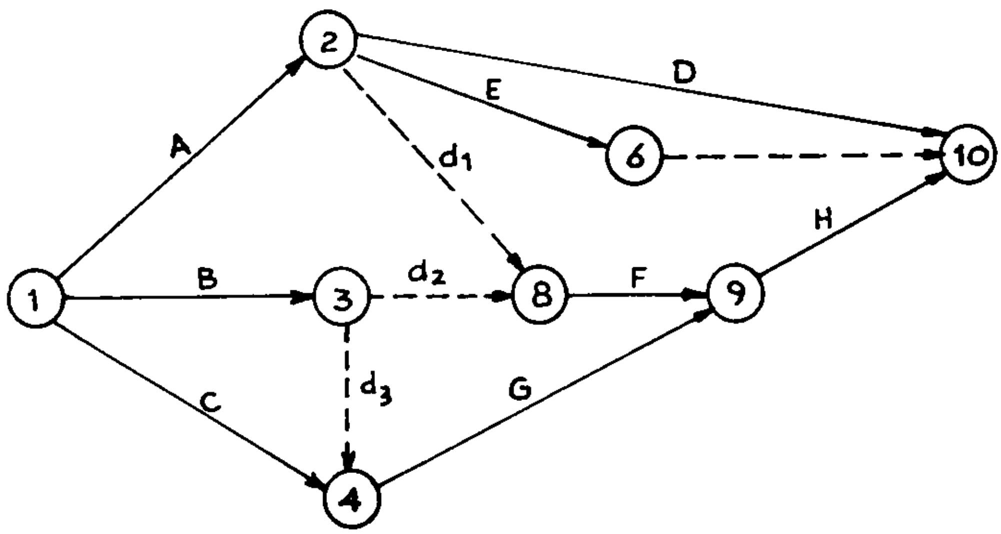


Fig. 11

After renumbering the nodes we have the final network as shown in Figure 12.



5. CONCLUSION

In this paper we have proposed an approximate algorithm for creating Activity-On-Arrow (AOA) networks for projects. The algorithm has been described in Section 3 and is explained with an example in Section 4. A simulator for generating test problems is being developed. Once ready it will be used to create test problem to examine the proposed algorithm.

REFRENCES

1. Corneil, D.G., C.C. Gotlieb, and Y.M. Lee, "Minimal Event-Node Network of Project Precedence Relations," Comm. ACM, Vol. 16, May 1973, pp. 296-298.

2. Dimsdale, D., "Computer Construction of Minimal Project Network," IBM Systems Journal, Vol. 2, March 1963, pp. 24-36.

3. Fisher, A.C., D.S. Liebman and G.L. Neimhauser, "Computer Construction of Project Networks," Comm. ACM, Vol. 11, July 1968, pp 493-497.

4. Krishnamoorthy, M.S. and N. Deo, "Complexity of the Minimum-Dummy-Activities Problem in an Pert Network," Networks, Vol. 9, 1979, pp. 189-194.

5. Syslo, M.M., "Optimal Constructions of Event-Node Networks," RAIRO Recherche Operationnelle, Vol. 15, 1981, pp. 241-260.

6. Syslo, M.M., "On the Computational Complexity of the Minimum-Dummy-Activities Problem in a PERT Network," Networks, Vol. 14, 1984, pp.37-45.