International Journal of Advanced Computational Engineering and Networking, ISSN(p): 2320-2106, ISSN(e): 2321-2063
Volume-6, Issue-12, Dec.-2018, http://iraj.in

A MODIFIED HEURISTIC APPROACH FOR CONSTRUCTING AOA NETWORKS

[1]RAMI A. MAHER, [2]HAMZA AL-SAFADI

[1,2]Isra University, Amman, Jordan
E-mail: [1]rami.maher@iu.edu.jo, [2]safadi007@ymail.com

Abstract - This paper introduces a heuristic algorithm, which focuses on constructing a unique AOA network for any project from its activities, and immediate predecessor constraints list. The algorithm guarantees a minimum number of dummy nodes even when the project involves multi-critical paths. The proposed algorithm can be performed manually as well as automatically. The final output of the algorithm is to determine two network matrices that can be used directly for optimal project scheduling. The algorithm is explained through a detailed example of the process of converting any immediate predecessor list to AOA networks.

I. INTRODUCTION

There are two available representation methods for planning and scheduling project networks that are activities on arcs AOA and activities on nodes AON. Both AOA and AON network's diagram are widely used in project planning and scheduling, but they have some fundamental different in characteristics and representation [1]. AOA network represents project activities by arcs, which have specific length depending on the duration of the activity where, a node is an event used to represent activity end and separate the activity (an emanating arc) from each of its immediate predecessors (an entering arc). The precedence relationships between the activities can thereby by the sequencing of the arcs. On the other hand, AON networks represent the project activities by nodes containing specific information about each activity such as name and duration of the activity. Arcs in AON are only used to determine the precedence relationship between the activities. Each emanating, and entering arcs from the activity node represent the immediate predecessor and successor activities.

Choosing between these two types is based on the individual project requirement. There are several differences between AOA and AON [2, 3]. Most of these differences address that the AON is more preferable for project management. In addition, AOA has the problem of adding dummy activities. In spite of this, when the AOA is represented mathematically by corresponding network matrices, it becomes very suitable for optimizing the project scheduling. These matrices are easily and systematically created from AOA networks, especially for the large-scale and mega project [5, 6]. Researchers contribute in this field in different ways. In section V, these contributions are compared to the proposed heuristic algorithm.

The first heuristic algorithm to generate and construct AOA was introduced by Bernard Dimsdale, who discussed in his research a computer procedure to increasing overall efficiency in using existing programs by obtaining networks with certain minimal properties [7]. In 1968, Fisher and Liebman proposed heuristic algorithm used topologically order to create the minimial number of dummy activities and nodes for any given precedence relations [8]. In 1970, D.G. Corneil proposed a procedure for constructing a event-node network to represent a set of precedence relations, he claimed that his algorithm produce the optimal number of nodes and arcs [9].
In 1990s Herroelen design a procedure called a random activity network generator to generate networks with various sizes and structure, in addition to, provide a parameter for testing the accuracy and efficiency[10]. Elmaghraby and Herroelen offered a methodology and complete software to generate project networks and provide a complexity index [11]. Later in 2007, Cohen proposed an efficient algorithm that ensures to create a unique AOA network for a given precedence table[12]. In 2012, researchers develop an algorithm that depends on the graph theory to create the AOA network [13].
It is known that constructing AOA network suffers the addition of dummy activities and the additional nodes. In general, a minimum number of dummy activities and node will save the computation time. Thus, the main objective of this paper is to propose an algorithm that constructs a unique AOA network for any project from activity's list and its immediate predecessors, which guarantee a minimum number of dummy activities and nodes, including in a network. The proposed algorithm can be simply used manually as well as automatically to obtain the network matrices.

II. ALGORITHM DESCRIPTION

The logic of the algorithm can be described in the following points as follows:

- Each activity in AOA network has a specific node represent the start time; therefore, the first stage of the algorithm focusing on the

constructing a minimum number of start nodes for all project activities.

- Combination of the activities decreases the number of the required node to represent the project activities.
- Considering the rules to draw AOA networks, there are constraints on the flexibility of drawing, especially when some activities playing a part in more than one unique precedence constraint.
- Dummy activities are needed when there is more one start node or end node for the same activity.
- Dummy activities are needed when there is more one start node or end node for the same activity.
- The algorithm establishes the end node for each activity based on the sequencing of the activities.
- Determine the activity-network matrices.

III. DETAILS OF ALGORITHM STEPS

The basic concept of the algorithm is to establish a unique start and end node for each activity, and add a minimum number of the needed dummy activities to complete the network considering the correct logical precedence relationships between the project activities. Mainly the proposed algorithm consists of the four stages, as follows.

First: Constructing the start node for each activity

Let us number the given columns of activity names, and precedence relationships as first and second columns respectively. In this stage, it is to construct the start node for each activity, which consists of adding two new columns (third and fourth). The third column contains unique immediate precedence constraints, and the fourth column consists of the number of these constraints. Steps of this stage can be performed as follows:

1. Produce the third column by duplication of the immediate precedence column (the second column).
2. Find the repetitions in the third column, if there are there repetitions, then they are removed and replaced with dash symbol (-).
3. Open the fourth column and starting from 10, and numbering the unique constraints sequentially. Constraints are replaced by the dash symbol (-) such to take the same number of the first appearance of the constraint.

In this stage, it should be noticed the following note; the third column must contain unique constraints only, and the number of the nodes needed to represent the project network equal at least to the number of the unique precedence constraints plus one.

Second: Identify necessary dummy activities

The second stage of the algorithm is to define the necessary dummy activities (if any) that required to complete the correct sequences and save the logical relationship between the project activities. This stage mainly formed from three check steps, which can guarantee the minimum number of dummy activities. These check steps are as follows:

1. In the second column, it is to find activities, which have one immediate predecessor. For these activities, no dummy activities have to be added.
2. Look again to the second column to find if there is more than one activity, which does not mention in the column. If so, there are two possible cases as follows:
    a. If these activities have the same precedence activities, then the number of the required dummies is equal to the number of these activities minus one.
    b. If these activities do not have the same precedence, then no dummy activities are required.
3. If there are activities, which have multiple precedence constraints in the second column, then there are two possible cases:
    a. If one or more activities have the same precedence activities that depend on each other, then the numbers of dummy activities are equal to the number of the activities minus one.
    b. If these activities do not have dependency relationships, then:
        i. If this constraint appears once, then there is no need to add dummy activities.
        ii. If this constraint appears once, then the number of the required dummy activities is equal to the number of recurring activities.

After performing these three check steps, the information about needed dummy activities must be added to the generated table as rows at the end of the table, these rows should be consisted of the required information such as start node and end node of the dummy activity.

Third: Constructing of the end node for each activity

This stage of the algorithm is to construct the end node for each activity, which consists of adding another two new columns. The fifth column contains the successor activities for each activity, and the sixth column contains the end node for each activity. Steps of this stage can be performed as follows:

1. Open the fifth column, for each activity find where it appears as precedence constraint.
2. Open sixth column, and look to column number three and find the number of the starting node for successor activity(s) from the fourth column.

This stage based on the idea of the common nodes between activities, which mean that the start node for the successor activities represents the same end node

for precedence activity. It should be noticed that there are activities do not have successor activity(s), that because it may be followed by dummy activities, or it may be connected to the end node of the project. In such case, additional node must be added to complete the correct generation and constructing of the network. Figure 1 shows a flow chart that illustrates algorithm stages and its detailed steps.

Fourth: Determine the Activity-Network Matrices

For optimization procedures, the project has to be described by two matrices, the activity-duration matrix T and the activity-name matrix A. In the former, the completion time of each activity is placed in the corresponding matrix entry. In the latter, a number replaces the activity name (activity A by 1, activity B by 2, and so on). Since, in project management network, the arcs are only towards the project end activity, then these two matrices are respectively defined by

$$T = \begin{cases} t_{ij} > 0; & i \neq j, \forall\, i,j \rightarrow \text{project activity} \\ 0; & i \neq j, \quad \forall\, i,j \rightarrow \text{dummy activity} \\ 0; & i = j \\ \infty; & \text{otherwise}, \forall\, i,j \rightarrow \text{non activity} \end{cases}$$

$$A = \begin{cases} a_{ij}; & i \neq j, \quad \forall i,j \rightarrow t_{ij} \geq 0 \\ 0; & elsewhere \end{cases}$$

where t_{ij} are the completion time of activity, and a_{ij} are numbers of the activity between the i[th] node and the j[th] node.

IV. A ILLUSTRATIVE EXAMPLE

To explain the algorithm steps, a simple example is used to illustrate the constructing of the AOA network from the precedence relationship table is introduced. In addition, it is to create the network matrices that are the input for any management process such as scheduling, allocation, etc. Table 1 provides the data of a 12-activity project; the duration of each activity is omitted, but are written letter in table 7.

The algorithm works on translating the list of the activities to an equivalent network that can be used project scheduling and control. In the first column, the actual project activities are listed, and the predecessor activities are listed in the second column number.

The algorithm starts by copying the contents of the second column into the third column, and replace the repetitions in the third column by (-); in the program, this symbol is replaced by some suitable number. For example, activity A, B, and C have the same precedence activity (None). The first occurrence remains in table where the other repetitions are replaced with (-). Other examples, can be noted in activities E and F, which have the same predecessor activity B. Table 2 illustrates the results of the first stage – step 1 and 2.

The next step in the first stage is to open and fill the fourth column, which consist of the number of unique constraints only appears in the third column. The (-) places in the third column take the same number of the first occurrence in the column number. Table 3 summarizes the results of the first stage. From this stage, minimum number of required nodes is at least equal to the number of unique constraints plus one.

The second stage is to define the necessary dummy activities, which are needed to complete the network. Second stage is formed from multiple check steps, first to check, if there are activities that have one immediate predecessor. In the considered example, activities A, B, C, D, E, F, G, H, and J have one immediate predecessor activity; therefore, they do not need a dummy activity.

Table 1 Activities List and their Immediate Precedence

1st Col.	2nd Col.
Act.	Immediate Predessecor
A	None
B	None
C	None
D	A
E	B
F	B
G	C
H	D
I	A , E
J	F
K	B , G
L	H , I , J , K

Table 2 Results of first stage (step 1 and 2)

1st Col.	2nd Col.	3rd Col.
Act.	Imm. Pre.	
A	None	None
B	None	-
C	None	-
D	A	A
E	B	B
F	B	-
G	C	C
H	D	D
I	A , E	A , E
J	F	F
K	B , G	B , G
L	H , I , J , K	H , I , J , K

Table 3 Results of first stage

1st Col.	2nd Col.	3rd Col.	4th Col.
Act.	Imm. Pre.		Start Node
A	None	None	1
B	None	-	1

C	None	-	1
D	A	A	2
E	B	B	3
F	B	-	3
G	C	C	4
H	D	D	5
I	A , E	A , E	6
J	F	F	7
K	B , G	B , G	8
L	H , I , J , K	H , I , J , K	9

The second check, is to find if there is more one activity does not mention in the second column. If there are no such activities, then no dummy activities must be added. Here in our example, activity L does not mention in the second column, but there are no other activities. Therefore, there is no need to add dummy activities.

The third check in this stage is to find the activities with multiple predecessor constraints. If there are such activities, then there are two possible cases. The first case, these activities have the same predecessor constraints, and one of these constraints depends on each other. The second case, these constraints do not have the dependency relationship (such as activity are I and activity K in our example). In such a case, there are two other possible cases, the first case is that the constraint appears more once time in the second column Activity A appears more once time and activity B also appears more once time. Therefore, dummy activities should be added before activities, I and K to complete the correct logical relationship between the activities. In the finish of this stage, the information about the needed dummy activities should be added. Table 4 shows the final results of the second stage after add dummy activities. Hence, the starting nodes are all found and numbered

Table 4 Results of second stage

1st Col.	2nd Col.	3rd Col.	4th Col.
Act.	Imm. Pre.		Start Node
A	None	None	1
B	None	-	1
C	None	-	1
D	A	A	2
E	B	B	3
F	B	-	3
G	C	C	4
H	D	D	5
I	A , E	A , E	6
J	F	F	7
K	B , G	B , G	8
L	H , I , J , K	H , I , J , K	9
D 1	A	-	2
D 2	B	-	3

The algorithm reaches the third stage, which is the constructing of the end node for each activity. This construction starts by open and fill the fifth column, which contain the successor activity(s) for each activity.

Filling the fifth column can be done by find where each activity appears as a precedence in the second column or not. In the considered example, activity A appears as a predecessor for three activities D, I, and the dummy activity D1. Therefore, the successor activities for activity A are activities D, I, and D1. It should be noticed that the successor activities for the added dummy activities will be the activities that required dummy activities, I and K. Table 5 demonstrates the results of step 1 from the third stage. It is remarkable to note that the total number of nodes becomes 10 (9 plus a finishing node) with 14 activities. Consequently, the network matrix is of order nine.

Table 5 Results of third stage (step 1)

1st Col.	2nd Col.	3rd Col.	4th Col.	5th Col.
Act.	Imm. Pre.		Start Node	Successor Act.
A	None	None	1	D
B	None	-	1	E, F
C	None	-	1	G
D	A	A	2	H
E	B	B	3	I
F	B	-	3	J
G	C	C	4	K
H	D	D	5	L
I	A, E	A, E	6	L
J	F	F	7	L
K	B, G	B, G	8	L
L	H, I, J, K	H, I, J, K	9	Finish
D 1	A	-	2	I
D 2	B	-	3	K

Step 2 of the second stage is to open the sixth column, which contain the number of the end node for each activity. The main idea in this step is that the start node for any activity represents the end node for a predecessor. Therefore, the end node will be taken from the start node for successor activities. In our example, activity D has a node number equal to 2, and it represents a successor for activity A. Therefore, the end node for activity A is 2. In addition, activities, E and F are successors for the activity B, and by looking into the fourth column it will be found that the activities, E and F have the same number, which is 3; thus, the numbering of the end node for activity B will be 3. Table 6 shows the results of the step 2 of the third stage.

Table 6 Results of third stage

1st Col.	2nd Col.	3rd Col.	4th Col.	5th Col.	6th Col.
Act.	Imm. Pre.		Start Node	Successor Act.	End Node
A	None	None	1	D	2
B	None	-	1	E, F	3
C	None	-	1	G	4
D	A	A	2	H	5
E	B	B	3	I	6
F	B	-	3	J	7
G	C	C	4	K	8
H	D	D	5	L	9
I	A, E	A, E	6	L	9
J	F	F	7	L	9
K	B, G	B, G	8	L	9
L	H, I, J, K	H, I, J, K	9	Finish	10
D 1	A	-	2	I	6
D 2	B	-	3	K	8

For activities that do not have successor activities, because these activities connected to the end node of the project, a new number for the nodes numbering must be added. For instance, activity L represents the project finish activity; thus, the finish node will be 10.

Table 7 shows the final results of applying the proposed heuristic algorithm for constructing the AOA.

Table 7 Activities duration and (from-to) representation

Activity	t	From – To
A	1	1 – 2
B	8	1 – 3
C	6	1 – 4
D	5	2 – 5
E	7	3 – 6
F	9	3 – 7
G	4	4 – 8
H	6	5 – 9
I	9	6 – 9
J	4	7 – 9
K	8	8 – 9
L	5	9 – 10
D 1	0	2 – 6
D 2	0	3 – 8

In the table, an assumed value of activity duration is listed too. Hence, the two matrices T and A are given by the following (10×10) matrices; number of nodes is 10. Obviously, the dummy activities have zero entries in the T matrix.

$$T = \begin{bmatrix} 0 & 1 & 8 & 6 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty & 5 & 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty & 7 & 9 & 0 & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & \infty & \infty & 4 & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 & \infty & \infty & \infty & 6 & \infty \\ \infty & \infty & \infty & \infty & \infty & 0 & \infty & \infty & 9 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & 0 & \infty & 4 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 8 & \infty \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 & 5 \\ \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & \infty & 0 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 13 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 6 & 14 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 7 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

CONCLUSIONS

This paper introduces a heuristic approach for generating AOA networks focusing on minimizing the number of dummy activities and saving the logical relationships between the project activities. The proposed algorithm ensures that the generated AOA network is unique for any given precedence table. The basic concept of the algorithm is to establish unique start and end node for each activity with a minimum number of needed dummy activities considering the precedence logical relationship between the activities. It is summarized assigning numbers at the start and end node for all activities. The algorithm formed mainly from four stages that are; construct the start node, identify necessary dummy activities and construct the end node. Future research could examine the algorithm on various complexities of the precedence lists. And it will be important to measure the complexity index for algorithm stages. Moreover, the algorithm is ready programmed to be augmented with any optimization process.

REFERENCES

[1] Lewis, J. P," Project Planning, Scheduling and Control", 4E: McGraw-Hill Pub. Co., 2005
[2] Demeulemeester, E., Herroelen, W., "International series in operations research and management science", Vol. 49, 2002
[3] Elmaghraby, S., Kamburowski, J., "On project representation and activity floats", Arabian Journal for Science and Engineering, 15(4 B), pp. 627-637, 1990
[4] Amer m. Al-Qnahrah, "Optimization of Multi-Resource Alocation in Large-Scale Project Management", M.Sc. thesis, Isra University-Amman, April 2015
[5] Maher F. Yousif, "Multi-Project Scheduling with Limited Resources Management in Construction Industry", M.Sc. thesis, Isra University-Amman, August 2017
[6] Dimsdale, B., "Computer construction of minimal project networks", IBM systems journal, 2(1), pp .24-36, 1963
[7] Fisher, A., " Computer construction of project networks", Communications of the ACM, 11(7), pp. 493-497, 1968

[8] Corneil, D. G., Gotlieb, C., Lee, Y., "Minimal event-node network of project precedence relations", Communications of the ACM, 16(5), pp. 296-298, 1973
[9] Demeulemeester, E., Dodin, B., Herroelen, W., "A random activity network generator", Operations research, 41(5), pp. 972-980, 1993
[10] Agrawal, M., Elmaghraby, S. E., & Herroelen, W. S. "DAGEN: A generator of test sets for project activity nets", European Journal of Operational Research, 90(2), pp. 376-382, 1969
[11] Cohen, Y., & Sadeh, A., "A new approach for constructing and generating AOA networks". Journal of Engineering, Computing and Architecture, 1(1), pp. 1-13, 2007
[12] Nasser Eddine Mouhoub, Samir Akrouf,"Generating PERT Network with Temporal Constraints", Studia Univ. Babes-Bolyal Informatica, Vol. LVII, No.4, 2012

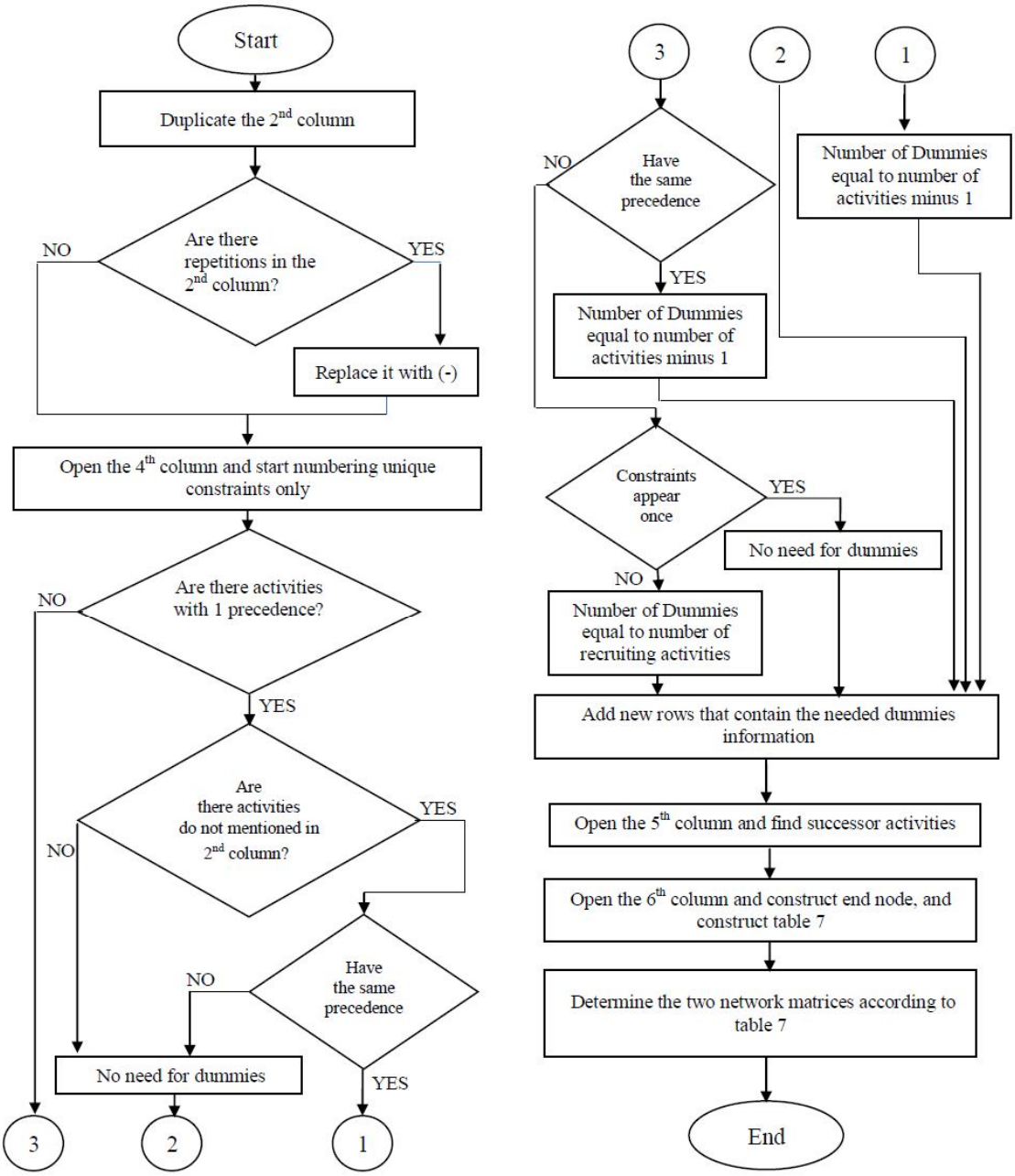

Figure 1 Heuristic algorithm to generate AOA

★★★