A New Approach for Constructing and Generating AOA Networks

Yuval Cohen[1]
Department of Management & Economics, The Open University of Israel,
108 Rabutzki Street, P.O.Box 808, Raanana 34104, Israel
yuvalco@openu.ac.il

Arik Sadeh
Department of Technology Management, Holon Academic Institute of Technology,
Golomb 52, Holon, Israel,
sadeh@hit.ac.il

[1] Corresponding author

A New Approach for Constructing and Generating AOA Networks

Abstract
A significant drawback of Activities on Arcs (AOA) networks, is having several different possible networks describing the same project. In contrast, the Activities on Nodes (AON) representation is unique. Having both AOA and AON networks of a project is an advantage since some planning and optimisation techniques strictly require AOA format while others require AON format. While previous AOA research has been focusing on minimizing the number of dummy arcs, this paper focuses on generating a meaningful consistent network. Each AOA node in the proposed technique is related to a specific row in the immediate predecessors table: its incoming arcs represent the immediate predecessors, and its outgoing arcs represent the activity/ies possessing this set of precedence constraints. The proposed algorithm generates a unique AOA network from a list of precedence constraints. While it does not minimize the number of dummy arcs the efficiency in dummies is a by-product of this approach. Two equivalent mathematical expressions are developed for the number of generated AOA dummies of a given precedence table. The paper also presents some linear formulations that could be directly derived from an AOA net. The algorithm is explained through a detailed example that starts with an immediate predecessors table and ends with an AOA network.

One line summary:
Method for constructing a unique AOA net with a node for each precedence constraint of its corresponding AON network (yielding small number of dummy arcs).

Keywords: Project scheduling, Network design, Network generation, AOA network, Activities on Arcs.

Journal of Engineering, Computing and Architecture

1. Introduction

Project networks for planning and scheduling are based on two representation types: Activities on Arcs (AOA) and Activities on Nodes (AON). Each representation type (AOA or AON) has desirable features that are unique to their type [16] (see next section for a comparison of AOA to AON). For example, some advantages of Activities on Arcs (AOA) networks for project network planning are: (1) each AOA arc has its origin and destination node, conforming with a "from-to" matrix (incidence matrix), or with variables in a node arc matrix. (2) AOA networks could be arranged in a total unimodular "node-arc" matrix that ensures an integer solution for integer data. (3) AOA is suitable for some popular formulations that use pairwise subtractions of AOA node variables.

While the construction of AON is relatively easy, AOA networks may contain dummy arcs and their construction is much more problematic. However, due to their importance, AOA networks have been drawing many research trials to generate AOA networks with minimal number of dummy arcs. Examples of dummy minimization heuristic methods can be found as early as in [7, 11, 12]. In the 1970's Corneil et. al. [4], claimed to have an optimal algorithm, but Syslo [20] disproved their claim. Krishnamoorty and Deo [13] showed that finding the minimum dummy problem is *NP-hard*. In the 1980's [19, 20, 21] offered more heuristics. In the 1990's [1, 5] proposed AOA network-generating methods but they did not consider the redundancy and other problems brought up by dummy activities, and therefore, are considerably impaired. A different direction was opened when Elmaghraby and Herroelen [8] developed a complexity index as a measurement tool for the complexity in activity networks. Kamburowski et. al. [14, 15] developed a method that generates minimal complexity-index AOA networks. Note however, that a single AON network may be represented by several different AOA networks with the same minimal complexity index.

The meaning of an AOA node in all of the above papers is an event in time, that does not necessarily correspond to anything expressed by the AON of the same project. Thus, the resulted net is convoluted and it is difficult (and sometimes impossible) to recover the original immediate precedence constraints.

The approach in this paper is different. We propose an AOA network in which each node corresponds to a precedence constraint. For this network, minimizing the dummy arcs is much easier and is an inherent part of the algorithm presented below. Although the algorithm logic is self evident, we still verified and validated it by testing it on a variety of different networks.

The proposed algorithm enables both a systematic translation of AON network to AOA network (and back to AON) and a systematic construction process of AOA networks from the corresponding immediate predecessor tables.

2. AOA vs. AON Network Representation

The first step in constructing an AOA network is to list the immediate predecessors of each activity and to sequence the activities in a table according to the precedence constraints. The resulting precedence table, corresponds directly to an Activities on Nodes (AON) network diagram [16]. In an AON diagram, the activities are represented on nodes and the precedence constraints by the arcs connecting the nodes. Therefore, in AON the number of nodes is the number of activities, and the number of arcs is the number of immediate predecessors. AON representation is natural and does not contain artificial elements on the network.

In contrast, AOA nets are characterized by dummy arcs and implicit precedence relations. As a result each project can have multiple AOA representations [10]. The different representations have actual and undesirable effect on the slack measures (time flexibility measures) of certain activities [6, 9].

On the other hand, the structure of the AOA network is much more suitable for certain analytical techniques and optimization formulations. For example, AOA activities correspond to entries in a matrix of origin to destination which in turn suites some linear programming formulations. Most of the network optimization techniques are based on either

node arc formulations or origin-destination matrix formulations (i.e., AOA representation and not AON representation). Also, most linear programming formulations for finding the critical path or "crashing" critical activities are based on AOA. Another kind of formulations utilize subtractions of AOA node variables to express the arc (activity) value.
The following are some examples of AOA based formulations:

I. A time crashing formulation taken from [2, p. 181]:

Define: *i,j* as pair of AOA nodes; t_{ij}=time for activity (*i,j*); *Sij*=cost/time-unit for activity (*i,j*).
Further define: M_{ij}=Crash duration for (*i,j*); N_{ij}=Normal duration for (*i,j*); E_i=time of node *i*; *O*= Overhead $ per time unit; *L*= index of last node; *T*=maximal specified time for the project.

Maximize:

Subject to:

(1) $M_{ij} \leq t_{ij} \leq N_{ij}$ *for all pairs (i,j)* $(\forall i,j)$
(2) $E_i + t_{ij} - E_j \leq 0$ *for all pairs (i,j)* $(\forall i,j)$
(3) $E_L \leq T$

II. Formulation for determination of the critical path from [6, page 112].

Define: *i* and *j* as indexes of AOA nodes, A=the set of all activities, *n*=last node, d_{ij}=AOA activity duration; P_j=set of predecessor nodes of node *j*; S_j=set of successor nodes of node *j*; x_{ij}= the quantity flowing on arc (*i,j*) from node *i* to node *j*. The formulation is given by:

Max

Subject to:
(1)

(2)

for *j*=2,3,4,5…,n-1

(3)
(4) All $x_{ij}>0$ for all $(ij) \in A$

III. AOA critical path formulation [17]:

Where $T_{i,j}$ is the time of an AOA activity that starts at node *i* and finishes at node *j*. T_i is the time for completing all the activities that lead to node *i*.
While small AOA networks can be easily drawn by hand, the construction of large AOA network is a complex task that requires time, rigor, and attention to details. An AOA construction algorithm can save a great deal of work and mistakes. Moreover, in many cases the user would like to go back and forth between a AON representation and a AOA representations. The logic of the following algorithm can make this desire a reality.

3. The Algorithm Logic

This section briefly describes the logic of the algorithm that generates a AOA network from a corresponding AON net or a precedence table. A more elaborate discussion could be found in [3]. However, the logic and its details are best explained in section 6.
The first step of the algorithm presented below is almost trivial: to build a simple precedence table, that lists the immediate predecessors (activities) for each activity, and to transform it into an "Activities on Nodes" network. For *n* tasks, the complexity of the first step is $O(n^2)$
The second step identifies the final AOA nodes. Each AOA node corresponds to a unique precedence constraint. Each node in AOA marks the starting event of the arcs that emanates from it. Thus, if two or more activities have the same precedence constraint, their arcs can emanate from the same node (which corresponds to that constraint). Combining the starting point of all the arcs that have the same precedence constraint, into one node, helps in minimizing the number of nodes in the AOA network. In the second step the algorithm generates a list of unique precedence constraints, each of which becomes a node in the final AOA diagram (including the start node for

activities with no predecessors). Finally, if more then one activity does not have any successor, a dummy node should be added symbolizing the end of the project.

The third step consists of identifying necessary dummies. A dummy arc is necessary whenever an activity is playing part in more than one unique precedence constraint (from step 2). The reason for needing the dummy arc is having more than one end node (each node represents the different precedence constraints of its successors). Thus, the algorithm's third step is to identify all these necessary dummies.

After the third step we have all the AOA nodes (from step 2) and all the AOA arcs (an arc for each activity and dummy arcs from step 3). All that remains to be done is to assign each start and finish pairs of nodes to their corresponding arcs. The rest of the algorithm steps achieve this end. In addition, giving the activities new names according to a 'from-to' scheme facilitates the formulation of the optimization problem and is also part of the algorithm. The algorithm contains a total of seven steps. The results of each step are summarized in a table as shown in the next section. Step seven ends with a table suitable for forming an "Activities on Arcs" network.

The seven steps and their complexities are as follows:

(1) Construct immediate predecessors table, $O(n^2)$; (2) Find a list of unique precedence relations and number them as the AOA nodes, $O(n)$; (3) Find necessary dummy activities, $O(n)$; (4) Add dummy activities to the table, $O(n)$; (5) Associate each AOA node with its incoming arcs, $O(n^2)$; (6) Associate each node with its outgoing arcs, $O(n)$; (7) Mark the arcs (activities) with indexes by their start and finish nodes, $O(n)$.

From the above the overall complexity of the algorithm is $O(n^2)$.

4. The AON Case Study

The example detailed below will assist in illustrating the algorithm. The example is based on a construction project of casting concrete. The data is provided in table 1, and an illustration of the data is given in figure 1 using AON representation.

Table 1. An Example of Activities and their Precedence Constraints

The Activity	Short name	Immediate Predecessors
A. Get steel rods and beams	Get steel	None
B. Get wooden rods and beams	Get wooden rods	None
C. Get beach sand	Get beach sand	None
D. Get concrete-powder	Get concrete-powder	None
E. Get tank for blending the concrete	Get blending tank	None
F. Prepare the casting skeleton as a grid of steel rods attached by wires and welding.	Prepare skeleton	A
G. Bend the steel beams for frame support	Bend the steel beams	A
H. Construct a concrete casting frame from wooden beams	Construct casting frame	B
I. Connect (with a pipe) a water hose to the mixing tank	Connect water	E
J. Mix concrete powder, sand and water in the tank	Mix concrete	C,D,I
K. Cover the molding frame with plywood boards	Cover frame with boards	H
L. Attach the internal skeleton of steel rods to the external molding frame	Attach skeleton to frame	F,H
M. Support the frame with steel beams	Steel beams frame support	G,H
N. Support the plywood boards with steel beams	Steel beams boards' support	G,K
O. Cast the concrete mix into the frame	Cast the concrete	J,L,M,N

5. Details of the Algorithm Steps

The steps of the algorithm are detailed below and summarized as follows: the first three steps are summarized in table 2, steps four and five are summarized in table 3 and the last steps are summarized in table 4. It is recommended to trace the algorithm at the end of each consecutive step, by comparing the step column/s and previous step column/s.

Step 1: Construct Immediate Predecessors Column

Prepare a column with the immediate predecessors of each activity. For example, see column 1 in table 2.

Step 2: Identify Activities with Identical Constraints (these activities will turn into Arcs emanating from the same node.

5.2.1 Duplicate the immediate predecessor column (column 1) into column 2.

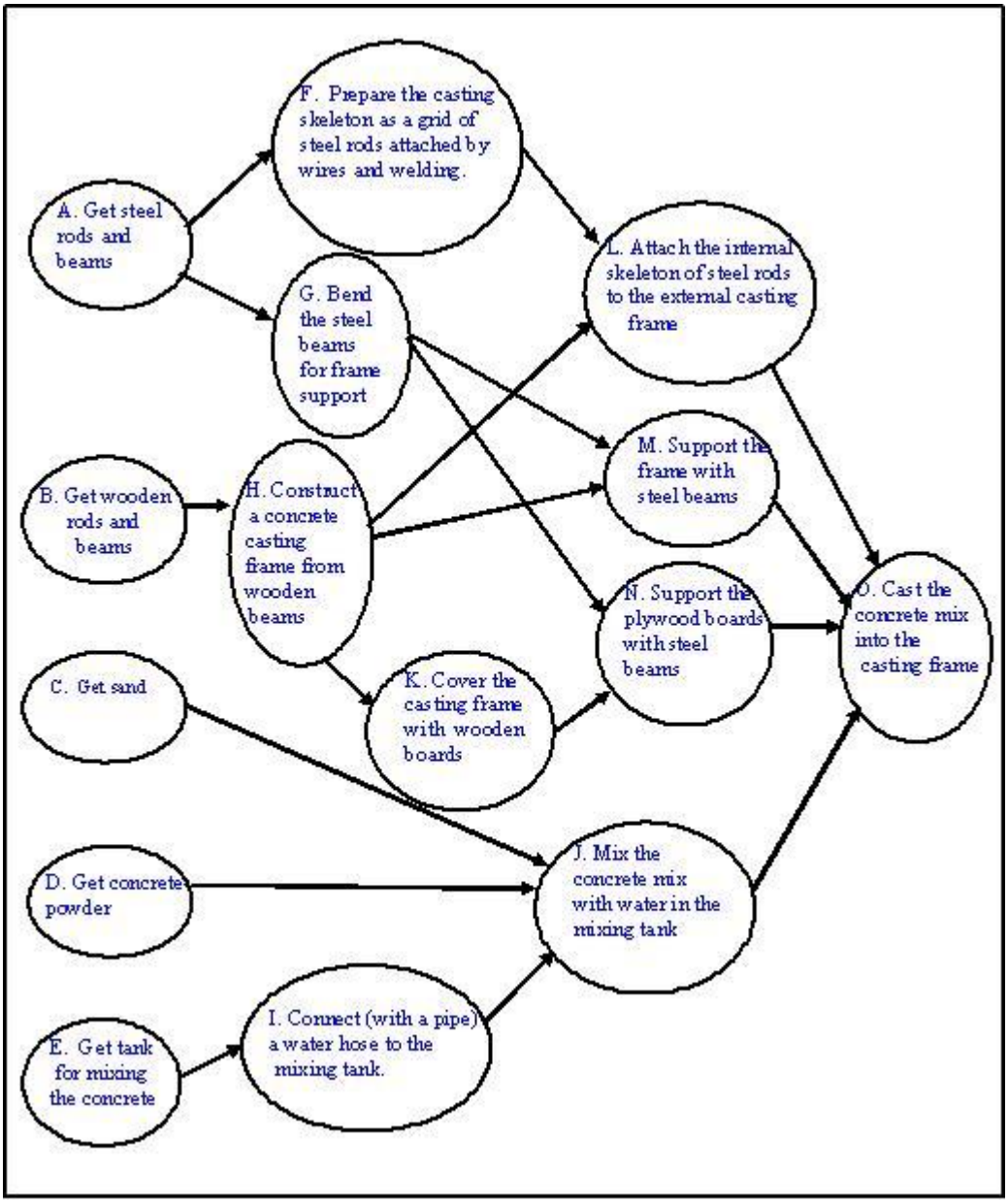

Figure 1. AON network for the precedence relations shown in Table 1.

5.2.2 Go over column 2 and find repeating precedence constraints. For each repeating constraint insert the mark "-"in column 2, blocking the entry of the repeating constraint.
We can see in column 2 that the repeating constraints are the null constraint (preceding activities A through E) and activity A (preceding activities F and G). Therefore, the arcs of activities A through E will originate from node 0 (in the AOA net) and the arcs of activities F and G will originate from the end node of activity A. Since column 2 must have a unique list with no repetitions, the repeating occurrences at rows B through E are blocked. The same is true for the row of activity G in column 2.
5.2.3 Open a third column (column 3) and number the unique precedence constraints sequentially. Repetitions get the same number as the first occurrence. Table 2 lists ten (0 through 9) unique precedence constraints corresponding to AOA nodes. Thus, we already know that together with the *end* node, the AOA in the example cannot have less then 10+1=11 nodes.

Step 3: Identify Necessary Dummy Arcs
5.3.1 Scan column 2 of table 2 (from step 2) and list activities that appear more than once in that column. For example, list activity G since it appears as a predecessor of activities M and N (note that activities M and N have different precedence constraints). Also, list activity H as it precedes activities K, L and M. Since all non-blocked entries are unique constraints, each of them represents an AOA node. The repeating activities (i.e., G, H) are

precedents to two or more AOA nodes. Since an arc can only point to one of these nodes, there is a need for dummy arcs pointing to the other nodes.

5.3.2 If one of the repetitions found in 3.1 above is a single activity, no dummy is needed for this repetition/constraint. The end node of the single predecessor activity becomes the starting point of the successor. For example, H is the only immediate predecessor of activity K. Thus, the arrow of activity K can originate at the end node of activity H, and there is no need for a dummy in this case. However, as H is a part of two other precedence relations, dummies are necessary.

5.3.3 Repetitions not conforming to 3.2 correspond to dummies. Rename these dummies with letters. For example, activity H is a predecessor of activities L and M. However, H is not the only predecessor of L or of M. Thus, both repetitions (L, M) do not conform to 3.2 and therefore, there is a need to use **two** dummy arcs. In column 3, the two appearances of H are numbered H1 and H2 to designate them as dummies. Another example is activity G that precedes activities M and N. Both repetitions do not conform to 3.2 and therefore are renumbered as G1 and G2 to designate them as dummies.

The first 3 steps of the algorithm using the example are summarized in table 2. Activities are marked by letters, and nodes are marked by numbers.

Table 2: Summary of the first three steps of the algorithm

	Step 1	Step 2		Step 3
Activity	Column 1: Immediate Predecessor Activities	Column 2: Unique Combinations Of immediate Predecessors	Column 3: Unique Combinations Numbers =AOA starting Node #	Column 4: Dummy Activities Identification (based on Column 2)
A	None	None	0	None
B	None	-	0	-
C	None	-	0	-
D	None	-	0	-
E	None	-	0	-
F	A	A	1	A
G	A	-	1	-
H	B	B	2	B
I	E	E	3	E
J	C,D,I	C,D,I	4	C,D,I
K	H	H	5	H
L	F,H	F,H	6	F,H1
M	G,H	G,H	7	G1,H2
N	G,K	G,K	8	G2,K
O	J,L,M,N	J,L,M,N	9	J,L,M,N

- Intentionally blocked entry

Step 4: Add Rows and Information for Dummy Arcs

For each dummy activity (identified in step 3), add a row at the end of table 2. Fill this row with the name of the dummy activity and write the real activity as the predecessor (for example, add a new row for activity G1, and make G its predecessor). Column 1 of table 3 depicts the result of step 4.

Step 5: Associate Activities with their End Nodes

In this stage, we accept the work column as the new immediate predecessor column.

Steps 3 and 4 were devoted to solving the following problem: Whenever an activity is part of two or more different constraints, an arc or arrow (that points to only one node) is not enough for its representation. Thus, additional dummy arcs are needed to model this activity.
Thus, after steps 3 and 4 each activity has at most one non-dummy succeeding node (if there had originally been other nodes, they were assigned dummy arcs). From step 2 we know the starting AOA node of each activity (table 2, column 3). Since each activity has its own immediate predecessors, each starting AOA node serve also as the end node of its immediate predecessors. So for each activity we add two new columns: a successors column (column 3, table 3), and a successor start node column (column 4, table 3). The successors starting node is also the end node of the activity's arc.

Finding successors (table 3, column 3)

For each activity, scan column 1 in table 3 and find where the activity appears as an immediate predecessor. This activity is illustrated by arrows in table 3.
If an activity still misses a successor, it is either because its end node is followed by dummies only, or it is the project's end node. For these missing end node entries add new node numbers. For example, since activity G is followed by dummies only, it needs an artificial node. Therefore, node 10 is added (see table 3, column 4). Also, node **11** is added for the project end node (see table 3, column 4).

Finding the successor start node (which is the activity end node - table 3, column 4)

The start node of any non-dummy activity (successor activities included) is already in column 2 of table 3 (taken from table 2, column 3). Once a successor has been found, its start node must be the end node of the predecessor arc. Looking up in column 2 of table 3 at the successor's row gives the successor's start node. This start node is the end node of the predecessor activity, which is the activity of the current row. Thus, we go sequentially over all the rows and find their corresponding end nodes in that fashion.

Table 3. Summary of steps 4 and 5. (The lookup procedure of step 5 is illustrated with arrows for activity A.)

	Step 4	Step 5		
Activity	Column 1: Immediate Predecessor Activities	Column 2: AOA starting Node # (Table 2, Column 3)	Column 3: Succeeding Activities	Column 4: Successors Start Node = AOA end node
A	None	0	F, G	1
B	None	0	H	2
C	None	0	J	4
D	None	0	J	4
E	None	0	I	3
F	A	1	L	6
G	A	1	G1, G2	**10**
H	B	2	K, H1, H2	5
I	E	3	J	4
J	C,D,I	4	O	9
K	H	5	N	8
L	F,H1	6	O	9
M	G1,H2	7	O	9
N	G2,K	8	O	9
O	J,L,M,N	9	End	**11**
G1 (dummy)	G		M	7
G2 (dummy)	G		N	8
H1 (dummy)	H		L	6
H2 (dummy)	H		M	7

- Intentionally blocked entry

Step 6: Associate Dummy Arcs with Their Start nodes

Each dummy arc represents the end of its associated non-dummy activity. Thus, the dummy arc should follow the end node of the real arc of its activity. All end nodes are identified in step 5. Therefore, step 5 is required before step 6 to enable the designation of an origin node to each dummy arc. To recapitulate, the start node of each dummy is the end node of its corresponding real activity. For example, dummy activity G1 starts where activity G ends. Since activity G ends at node 10, the arc of activity G1 starts at node 10. The bottom of Column 2 in table 4 depicts the results of step 6.

Example for step 6:

Dummy Activity	Real Activity	End Node of Real Activity	Start Node of the Dummy
G1		G	10 → 10
G2		G	10 → 10
H1		H	5 → 5
H2		H	5 → 5

Table 4: Summary of steps 6 and 7. This table is the basis for the AOA

	Step 4	Step 6	Step 7		
Activity	Column 1: Immediate Predecessor Activities	Column 2: AOA starting Node Addition of start Nodes for dummies	Column 3: AOA end nodes With step 7 Changes	Column 4: AOA Activity Names	Column 5: AOA Immediate Predecessor Activities
A	None	0	1	$A_{0,1}$	None
B	None	0	2	$A_{0,2}$	None
C	None	0	4	$A_{0,4}$	None
D	None	0	12	$A_{0,12}$	None
E	None	0	3	$A_{0,3}$	None
F	A	1	6	$A_{1,6}$	$A_{0,1}$
G	A	1	10	$A_{1,10}$	$A_{0,1}$
H	B	2	5	$A_{2,5}$	$A_{0,2}$
I	E	3	4	$A_{3,4}$	$A_{0,3}$
J	C,D,I	4	9	$A_{4,9}$	$A_{0,4}$; $A_{12,4;}$ $A_{3,4}$
K	H	5	8	$A_{5,8}$	$A_{2,5}$
L	F,H1	6	9	$A_{6,9}$	$A_{1,6}$; $A_{5,6}$
M	G1,H2	7	9	$A_{7,9}$	$A_{10,7}$; $A_{5,7}$
N	G2,K	8	9	$A_{8,9}$	$A_{10,8}$; $A_{5,8}$
O	J,L,M,N	9	11	$A_{9,11}$	$A_{4,9}$;$A_{6,9}$;$A_{7,9}$;$A_{8,9}$
G1 (dummy)	G	10	7	$A_{10,7}$	$A_{1,10}$
G2 (dummy)	G	10	8	$A_{10,8}$	$A_{1,10}$
H1 (dummy)	H	5	6	$A_{5,6}$	$A_{2,5}$
H2 (dummy)	H	5	7	$A_{5,7}$	$A_{2,5}$
D1 (dummy)	D	12*	4	$A_{12,4}$	$A_{0,12}$

* follows by dummies only

Figure 2 depicts the AOA network associated with table 4.

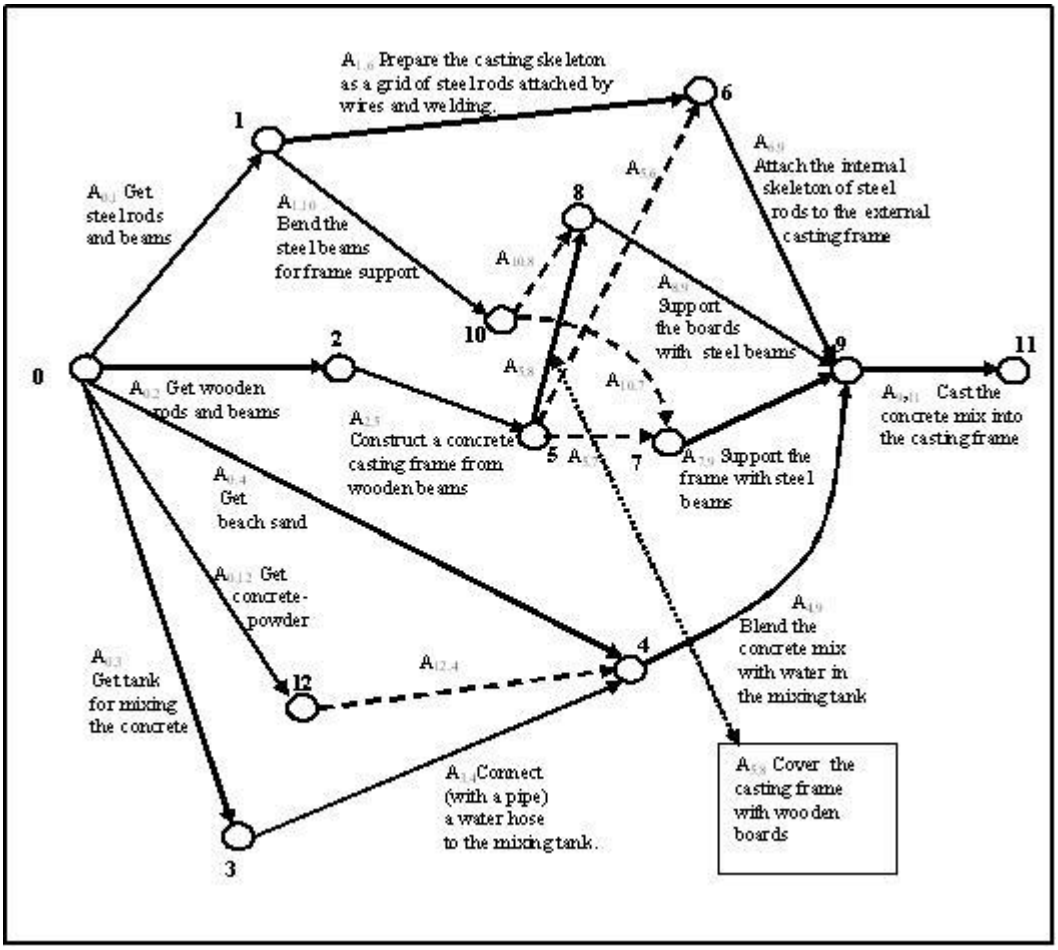

Figure 2. Activities on Arcs (AOA) Network for the "Concrete Casting" Project

Step 7: Check for necessity to add nodes and dummies to compensate for the effect of a single project start node, and a single project end node. Update the table with the new activity names.

It is a convention to have a start node for the start of a project. Hence, when a project net starts with several parallel arcs from the start node, some of these arcs may end at the same end node, creating a situation where different arcs have the same start and end nodes. For classic notations such as A_{ij} there is no way to differentiate between these arcs, and in addition this is not a tolerable format for some LP formulations. Therefore, dummy node and arc are added for such cases (until there are no two arcs with the same start and end nodes). For example, both activities C and D start at node 0 and end at node 4. Therefore, a dummy node 12 is added for activity 4 and a new dummy arc. Now activity 4 starts at node 0 but ends at node 12 (column 3 in table 4). Next, we connect an additional dummy arc from node 12 to node 4 (bottom of table 4).

The same situation can arise at the end of the project (since it is customary to have a single end node per project). Again, the solution is to add dummy nodes and arcs as necessary. These situations are easy to detect by scanning the columns of the start and end nodes. If the same pair appears more then once, then dummy node and arc are necessary for each repetition. At the end of this process each arc should have its unique start/end node combination.

After making sure that each arc has its own unique combination of start/end nodes, the arc names should refer to this combination. The name of each activity is changed to a letter and two indexes marking its start and end nodes (see column 4 table 4).

Table 4 is suitable for the construction of Activities on Arcs Network. It contain every AOA activity (including the dummy activities) and the start and end nodes associated with it.

6. Translating the Generated AOA to AON

The construction of AON network from its precedence table (where all the immediate predecessors of each activity are listed) is a trivial task. Thus, describing the steps that generate the precedence table from the generated AOA, is enough to supply all the information needed for generating an AON network. The steps that generate the precedence table from the generated AOA are as follows:

1. All real (non-dummy) activities are assigned AON node numbers.

2. Duplicate the AOA net with the new node number from step 1. In the duplicated net, give the dummy arcs the same number as their predecessors.

3. For each AOA activity, find and list the numbers of its immediate predecessors (numbers from step 2). The obtained list is the precedence constraint of the activity.
The output of step 3 is a precedence table. Thus, these three steps are enough to accumulate all the information needed for generating an AON network.

7. Characteristics of the generated AOA network

This section focuses on two main subjects: (1) The number of dummies, and (2) The correspondence between the precedence table and the generated AOA.
So far the method is presented in somewhat intuitive form. To add some formality to the discussion, some definitions along with propositions and their proofs are presented next:

Definitions:

N - the number of activities or AON nodes
P - the number of AON arcs (the number of immediate precedence relations)
X - set of all AON nodes having multiple emanating arcs
Y - subset of **X,** composed of nodes of **X** that all their emanating arcs terminate in nodes that have other incoming arcs (from other nodes).
Z - number of AON nodes that all their incoming arcs are from **Y**
A - the number of emanating arcs of the subset **Y** (*A*=2 in "Case B" of Figure 3).
d_i^+ - the outbound degree of node i (number of emanating arcs from node *i* AON)

D = $E(d^+)$ = - the average (expected value) outbound degree of a node in a AON network.
S - number of dummies needed for keeping a unique pair of nodes for activities artificially connected to the start or end nodes.

Proposition 1. The number of generated dummies in the generated AOA () is:
Number of Dummies = *S + P - N + A - Z* **(1)**

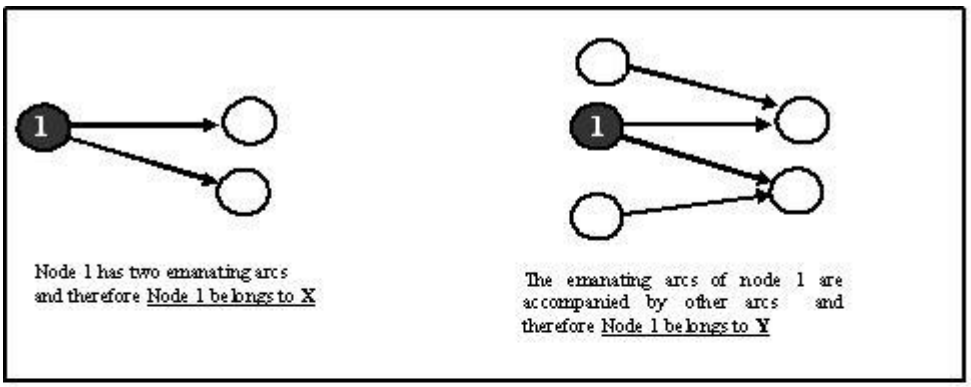

Figure 3. Illustration of examples of: (a) node belonging to set **X** and (b) a node belonging to **Y**.

Proof 1. *S* is added for correctly accounting for the addition of single start and end nodes. The number of arcs in the AON (*P*) is the number of pairwise precedence relationships each must be expressed by an arc in the AOA. If all *N* real activities would be used to express immediate precedence then the number of dummies would have been *P-N*. However, of the *N* real arcs, in (*A-Z*) cases the real activity could not be used directly to convey precedence (*Z* prevents duplicity). Thus, the number of "real" AOA arcs that are used as direct predecessors, is *N-A+Z*.
Since all AOA arcs have to convey all (*P*) pairwise precedence relationships, and since there are (*N-A*) real AOA arcs that convey immediate precedence relationship, the remaining (*P-(N-A+Z)*) = ***P - N + A -Z*** precedence relationships must be dummies.

Lemma. The number of generated dummies in the generated AOA is also:
Number of Dummies = $S + N\cdot(D-1)+A-Z$ (2)

Proof. Since the total number of AON arcs (P) is also the total number of AON outbound degrees: *we can substitute* $P = N\cdot D$, in $P - N + A - Z$ (which is, from proposition 1, the number of dummy arcs). The result is: $N\cdot D-N+A-Z = N\cdot(D-1)+A-Z$. Q.E.D.

Example (using figure 1 for AON and figure 2 for its corresponding AOA):
From figure 1 we have: N=15, $P=\Sigma d_i$=18, A=1 (activity G), D=18/15=1.2
Correctly adding a single start node yields: S=1
Thus, from proposition 1 we have: **Number_ of_dummies =$S+P-N+A-Z$ = 1+18-15+1 = 5**
And from proposition 2 we have: **Number_ of_dummies =$S+N\cdot(D-1)+A$ = 1+15(1.2-1)+1= 1+15·(0.2)+1 = 5**
This indeed is the number of dummy arcs in figure 2.

Proposition 2. In the generated AOA, each combination of: (1) AOA node *i*, and (2) a non-dummy arc emanating from the node *i*, corresponds to a row in the immediate predecessors table.

Proof 2. In the generated AOA, by definition, each non dummy arc corresponds to a unique activity. Also, focusing on the starting node of an activity *i*, the algorithm attaches (in step 5) the incoming arcs into node *i*, which correspond to the immediate predecessors of this activity, to node *i*. In case a dummy is a predecessor, it is directly connected to the end of its real activity.

One of the important by-products of proposition 2 is that the generated AOA net has the same early-start (*ES*) and late-finish (*LF*) times for each activity as in the corresponding AON net.
Also, Each of the AOA arcs (real or dummy) corresponds to a precedence relationship - that is an AON arc.

8. Conclusion
This paper presents a new approach for generating AOA networks: instead of trying to minimize the number of dummy arcs (and losing the difference between immediate and not immediate predecessors)
an AOA network construction algorithm that builds a network, in which the incoming arcs of each node correspond to the immediate precedence constraints of the nodes' outgoing arc/s.
The method ensures the generation of a unique AOA network for a given precedence table. While the algorithm does not minimize the number of dummy arcs, it is very efficient: the method adds dummies only for pointing at more then one precedence constraint. Any dummy removal destroys this immediate precedence logic, and any other dummy arrangement destroys the analogy to the precedence table. Expressions for the number of generated dummy arcs are developed. The new method presented here enables a systematic construction of a unique AOA network from a list of tasks and precedence constraints. Thus, it enables a systematic translation back and forth from an AON into an AOA network.

REFERENCES

[1] M. K. Agarawal, S. E. Elmaghraby, W. S. Herroelen, 1996. DAGEN a generator for test sets for project activity nets, *European Journal of Operational Research*, 90, 376-382.

[2] N. H. Ahuja, S. P. Dozzi, , S. M. Abourizk, , 1994. *Project Management Techniques in Planning and Controlling Construction Projects* (2nd Edition, John Wiley and Sons Inc.).

[3] Y. Cohen, 2003. A New Approach for Automated Construction of Activities on Arcs (AOA) Networks, *Research report - July 2003-1*, Deposited at the library, The Open University of Israel.

[4] D.G. Corneil, C.C. Gotlieb, and Y.M. Lee, 1973. Minimal event-node network of project precedence relations, *Communications of the SCM*, 16, 296-298.

[5] E. Demeulemeester, B. Dodin, and W. Herroelen, 1993. A random activity network generator, *Operations Research*, 41 (5), 972-980.

[6] E. Demeulemeester, L. W. Herroelen, and S. Willy, 2002. *Project Scheduling, A Research Handbook*, Kluwer Academic Publishing.

[7] D. Dimsdale, 1963. Computer construction of minimal project network, *IBM systems journal*, 2, 24-36.

[8] S. E. Elmaghraby, and W. S. Herroelen, 1980. On measurement of complexity in activity networks, *European Journal of Operational Research*, 5, 223-234.

[9] S. E. Elmaghraby, and J. Kamburowski, 1990. On project representation and activity floats, *Arabian Journal of Science and Engineering*, 15, 625-637.

[10] S. E. Elmaghraby, 1995. Activity nets: A guided tour through some recent developments, *European Journal of Operational Research*, 82, 383-408.

[11] M. Hayes, 1969. The role of activity precedence relations in node-oriented networks. *In H.J.M. Lombaers (ed.), Project planning by network analysis*, North-Holland Publishing Company, Amsterdam, 128-146.

[12] A.C. Fisher, D.S. Liberman, and G.L. Nemhauser, 1968. Computer construction of project networks, *Communications of the ACM*, 11, 493-497.

[13] M. S. Krishnamoorty, and N. Deon, 1979. Complexity of minimum-Dummy-Activities Problem in a Pert Network, *Networks*, 9, 189-194.

[14] J. D. Kamburowski, J. Michael, and M. Stallman , 1992. Optimal construction of Project Activity Networks, *Proceeding of the Annual Meeting of the Decision Sciences Institute*, San-Francisco, CA, 1424-1426.

[15] J. D. Kamburowski, J. Michael, and M. Stallman, 2000. Minimizing the Complexity of an Activity Network, *Networks*, 36 (1), 47-52.

[16] P. J. Lewis, 1995. *Project Planning, Scheduling and Control*, McGraw-Hill.

[17] S. Nahmias, 2001. Pr*oduction and Operations Analysis, 4th Edition*., McGraw-Hill Irwin.

[18] A. Shtub, J.Bard, S. Globerson, 1994. *Project Management: Engineering, Technology and Implementation*, Pearson Education.

[19] J. Spinard, 1980. The Minimum Dummy Task Problem, *Networks*, 16, 331-348.

[20] M..M. Syslo, 1981. Optimal Construction of Event-Node Networks, *RAIRO*, 15, 241-260.

[21] M..M. Syslo, 1984. On the Computational Complexity of the Minimum dummy Activities Problem in PERT Network, *Networks*, 14, 37-45.