

SAS/OR® PROVIDES BEST USE OF PERSONNEL

Major Doug McAllaster
US Army Personnel Command, Alexandria, VA

ABSTRACT

This paper provides an comprehensive introduction to one of the operations research tools in SAS/OR, PROC NETFLOW. Through an actual human resources application, this paper demonstrates the optimization capability of this powerful network flow programming procedure.

INTRODUCTION

The US Army is downsizing to meet mandated personnel end strengths. Senior Army leaders want to insure that the Army retains enough commissioned officers in each grade and career field so that the Army can fill its jobs at the conclusion of the drawdown. This paper describes how we use PROC NETFLOW to determine which commissioned officer career fields are under strength by grade. Once senior Army decision-makers know which career fields are critically short, they can protect those career fields from suffering further losses during the drawdown. That is, they can retain officers having shortage career fields and retire officers who do not have shortage career fields. This selective downsizing improves the Army's capability to fill its jobs. Retaining officers having required career fields insures that down-sizing is right-sizing.

MODEL THE PROBLEM

Operations researchers call the problem of matching people to jobs the assignment problem. We model the assignment problem with a supply and demand network. Figure one is a small portion of the assignment model under consideration. We represent the available supply of officers on the left and the required demand of jobs on the right. We call the supply and demand points nodes. We call the lines that connect nodes arcs. We want to know how much supply should move through each arc so as to fill the demand at least total cost.

Each node on the left represents a supply of officers having the same grade and career fields. The positive number to the left of each node is the number of officers having that combination of grade and career fields.

For example, the bottom node on figure one represents a supply of 31 officers at grade 4 who are qualified in career fields 25 and 54.

Each node on the right represents a demand of jobs requiring a particular grade and career field. The negative number to the right of each node is the number of jobs requiring that grade and career field combination. For example, the underlined demand node in the center of figure one indicates that there are 799 jobs at grade 4 in career field 25.

We represent a feasible assignment of officers to jobs with an arc from a supply node to a demand node whenever there is a match by grade and career field. Most Army officers are qualified in two career fields. Furthermore, officers can fill a job at one grade above their current grade, called upper substitution. Therefore, four arcs depart most supply nodes. For example, the officers from the bottom supply node 4-25-54 can be assigned to work in jobs requiring a 4-25, 4-54, 5-25, or 5-54. Note that each arc can be uniquely identified by its from and to nodes.

Note that figure one has a node with a # sign as its second career field. This node represents an inexhaustible supply of imaginary officers at that grade having that career field. We need this inexhaustible supply of officers in order to insure that the problem has a feasible solution, that is, that the model can fill all the jobs. Network flow programming enforces a conservation of flow constraint on each node. That is, we cannot use more officers than each nodes supplies and we must meet the demand of jobs at every node.

Figure two lists an arc for each feasible assignment from figure one. Note that each arc has an associated cost. Network flow programming requires that each arc have an associated cost in order to calculate the best feasible (optimal) solution. The optimization procedure then charges this cost for every unit of flow that moves through the arc from the supply to the demand.

The question mark represents the quantity of flow that moves through the arc in the optimal (minimum cost)

solution.

DEFINE THE PROBLEM

Next, we must define what we mean by the best solution. In this case, we define the best solution as the one that leaves the fewest jobs vacant. Because an imaginary officer cannot fill an actual job, our objective is to minimize the use of imaginary officers.

In order to accomplish this objective, we must develop a costing structure to force the assignment model to use all of the available officers before it uses any from the imaginary supply. This required costing structure follows. We must insure that the cost of an assignment from the available officers is always less expensive than an assignment from the imaginary supply. With this costing structure, the assignment model will use the available (inexpensive) officers before it uses the imaginary (expensive) ones.

CREATE THE SAS® DATA SETS

PROC NETFLOW requires two input SAS data sets: the `nodedata` and the `arcdata`. PROC NETFLOW produces an output SAS data set called the `arcout` that contains the solution to the problem. The following paragraphs discuss these three data sets. (The underscored variables are special ones that PROC NETFLOW looks for by default.)

The `nodedata` data set contains an observation for every node in the network flow model. There are only two required variables, `_node_` and `_supdem_`. `_Node_` must be a character variable that uniquely identifies a `_node_`. `_Supdem_` must be a numeric variable that gives either the supply or demand on each node. A node that provides supply to the network has a `_supdem_` greater than zero. A node that requires demand to the network has a `_supdem_` less than zero. Creating the `nodedata` is fairly straight forward. The SAS programmer must uniquely identify each node and assign it a non-zero supply or demand.

The `arcdata` data set contains an observation for every arc in the network flow model. There are only three important variables: `_from_`, `_to_`, and `_cost_`. The `_from_` and `_to_` are required character variables whose values must identify a `_node_` variable in the `nodedata`. Each arc must have a unique `_from_ to_` combination. `_Cost_` is a required numeric variable that assigns a cost for each unit that flows through the arc. Creating the `arcdata` is not so straight forward. The SAS programmer must use the DATA step to match the supply nodes

to the demand nodes and to assign a cost to each arc. Aster and Seidman explain a useful technique called match-crossing data sets in *Professional SAS Programming Secrets* that accomplishes this task. Although not required, the `_capac_` variable is often useful. `_Capac_` gives a maximum allowable flow through the arc. In this example, we set `_capac_` equal to one-third of the arc's demand for upper substitution arcs.

The `arcout` data set contains the optimal solution to the network flow model. It contains the same variables as the `arcdata` and more. `_Flow_` is the critical output variable. PROC NETFLOW calculates the `_flow_` through each arc so as to minimize the sum total of the `_flow_ times _cost_ product` for each arc. Furthermore, PROC NETFLOW enforces two conservation of flow constraints at every node. First, the flow out of a node cannot exceed its available supply. Second, the total flow into each demand node must meet the required demand. `Arcout` also lists the `_demand_` at each arc's `_to_ node` and the `_supply_` at each arc's `_from_ node`.

SOLVE THE PROBLEM

PROC NETFLOW solves the assignment problem. That is, PROC NETFLOW calculates the optimal flow of supply to demand through the network so as to fill the demand at the minimum total cost. It uses a variant of the revised primal simplex algorithm that exploits the network structure of the problem. PROC NETFLOW required fewer than 20 seconds to calculate the solution to this problem on a small work-station. This problem has 3373 arcs that connect 1478 supply nodes to 133 demand nodes.

IMPLEMENT THE SOLUTION

The top half of figure three shows a subset of the optimal solution. It shows how PROC NETFLOW filled the demand on the underlined demand node on figure one. Note that the algorithm often uses either all or none of a node's supply. This is not required but is a property of extreme point algorithms like the primal simplex method. Also note that the flow from node 3-25-00 to node 4-25 is at its capacity, one-third of the arc's demand.

The bottom half of figure three is an extract of one of the reports we provide to the Office of the Deputy Chief of Staff for Personnel (ODCSPER) at Department of Army in the pentagon. ODCSPER wants to know which career fields are critically short so that they can

insure that officers having those career fields are retained during the drawdown. Flow is the number of jobs that the Army can fill. Vacant is the number of jobs that the Army cannot fill. That is, vacant is the minimum number of imaginary officers required to fill the jobs. As a result of this analysis, the Army is retaining majors having career fields 18, 25, or 74.

CONCLUSION

Network flow programming is one of the most widely used optimization techniques in the field of operations research. Operations researchers can formulate many resource allocation and distribution problems using network models. The SAS System and SAS/OR provide the data management and optimization capability to solve these problems.

AUTHOR CONTACT

Contact Major Doug McAllaster by phone at (703) 325-5134, by electronic mail at mcallasd@hoffman-emh1.army.mil, or by letter mail at 115 South Royal St B34, Alexandria VA 22314.

REFERENCE

Aster, Rick and Seidman, Rhena (1991), *Professional SAS Programming Secrets*, New York, McGraw Hill, Inc.

TRADEMARKS

SAS and SAS/OR are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

FIGURE ONE -- ASSIGNMENT MODEL

SUPPLY of officers

DEMAND of jobs

	<u>_from_</u>	<u>_flow_ thru _arcs_</u>	<u>_to_</u>	
<u>_supdem_</u>	<u>_node_</u>		<u>_node_</u>	<u>_supdem_</u>
1124	3-25-00=====	\	---->3-25	-1494
		/-----	----->3-21	-1131
1	3-25-21=====	\	---->3-25	-1494
		\-----	----->4-21	-552
		/-----	----->3-25	-1494
11	3-25-41=====	\	---->3-41	-104
		\-----	----->4-41	-159
			\\	
40	4-25-##-----	>	4-25	-799
			/////	
456	4-25-00=====	/	---->5-25	-417
		/-----	----->4-48	-338
3	4-25-48=====	/	---->5-25	-417
		\-----	----->5-48	-320
		/-----	----->4-52	-43
1	4-25-52=====	/	---->5-25	-417
		\-----	----->5-52	-42
		/-----	----->4-53	-332
38	4-25-53=====	/	---->5-25	-417
		\-----	----->5-53	-193
		/-----	----->4-54	-459
31	4-25-54=====	/	---->5-25	-417
		\-----	----->5-54	-515

FIGURE TWO -- ASSIGNMENT PROBLEM

	<u>_from_</u>	<u>_cost_</u>	<u>_flow_</u>	<u>_to_</u>	
<u>_supdem_</u>	<u>_node_</u>			<u>_node_</u>	<u>_supdem_</u>
1124	3-25-00	1	?	3-25	-1494
1124	3-25-00	3	?	4-25	-799
1	3-25-21	2	?	3-21	-1131
1	3-25-21	1	?	3-25	-1494
1	3-25-21	4	?	4-21	-552
1	3-25-21	3	?	4-25	-799
11	3-25-41	1	?	3-25	-1494
11	3-25-41	2	?	3-41	-104
11	3-25-41	3	?	4-25	-799
11	3-25-41	4	?	4-41	-159
40	4-25-55	10	?	4-25	-799
456	4-25-00	1	?	4-25	-799
456	4-25-00	3	?	5-25	-417
3	4-25-48	1	?	4-25	-799
3	4-25-48	2	?	4-48	-338
3	4-25-48	3	?	5-25	-417
3	4-25-48	4	?	5-48	-320
1	4-25-52	1	?	4-25	-799
1	4-25-52	2	?	4-52	-43
1	4-25-52	3	?	5-25	-417
1	4-25-52	4	?	5-52	-42
38	4-25-53	1	?	4-25	-799
38	4-25-53	2	?	4-53	-332
38	4-25-53	3	?	5-25	-417
38	4-25-53	4	?	5-53	-193
31	4-25-54	1	?	4-25	-799
31	4-25-54	2	?	4-54	-459
31	4-25-54	3	?	5-25	-417
31	4-25-54	4	?	5-54	-515

FIGURE THREE -- ASSIGNMENT SOLUTION

supply	_from_	_cost_	_flow_	_to_	_demand_
1124	3-25-00	3	266	4-25	799
1	3-25-21	3	1	4-25	799
11	3-25-41	3	1	4-25	799
456	4-25-00	1	418	4-25	799
3	4-25-48	1	3	4-25	799
1	4-25-52	1	1	4-25	799
38	4-25-53	1	38	4-25	799
31	4-25-54	1	31	4-25	799
40	4-25-55	10	40	4-25	799
			=====		
			799		
to	_flow_	_demand_	_vacant_	fillrate	
2-25	595	627	32	0.94896	
3-18	390	561	171	0.69519	
3-25	1419	1494	75	0.94980	
4-18	230	254	24	0.90551	
4-25	759	799	40	0.94994	
5-25	396	417	21	0.94964	
5-74	122	126	4	0.96825	
6-25	187	199	12	0.93970	
6-74	34	36	2	0.94444	