Paper 241-2010

A Network Optimization solution using SAS/OR® tools for the Department of the Army Branching problem.

by
Michael C Grierson

ABSTRACT

This paper will demonstrate the use of SAS® and SAS/OR® to solve a long standing Army problem of assigning ROTC cadets to their initial basic branch (Infantry, Armor, etc). The paper starts with a problem statement, describes the problem as a network optimization and then shows model results after adding each of the constraints from the problem statement. The paper summarizes with comments about why a network optimization is a good solution for this type of problem. Finally, the paper makes the assertion that SAS® data manipulation and statistics procedures are an additional benefit provided by the SAS/OR® solution not found elsewhere.

INTRODUCTION

This paper shows how the Army could optimize the assignments of Reserve Officer Training Corp (ROTC) cadets to their initial basic branch in the Army using a network optimization. After an overview of the business problem is provided, we'll implement a solution using the NETFLOW procedure and repeat that network solution using the OPTMODEL procedure. The OPTMODEL implementation will be extended with additional constraints.

In this paper, we'll solve this problem one step at a time. Initially, we'll just assign the cadets to their branches based on their preferences and Army branch 'demand', then add constraints and assess the impact of these constraints.

PROBLEM DETAILS

The following facts apply to the ROTC assignment example used in this paper.

We start with a 'supply' of 2545 cadets.
Each cadet has 5 basic branch preferences, gender, and an Order of Merit Score (OMS) attribute.
The Army basic branches have a total of 2545 assignments (demands).
The Army cannot put females in combat arms branches (IN, AR, and FA).
The Army needs to proportionally distribute cadets based on gender, and OMS.
The objective of this model is to maximize cadet satisfaction while still meeting Army need.

The initial problem of assigning 2545 cadets to basic branches with maximum cadet satisfaction (a cadet is scored as satisfied if they get their 1st, 2nd or 3rd preference) will use the OMS values, cadet preferences, and the branch demands. Subsequently, constraints related to gender and OMS values will be included.

AN INITIAL MODEL

An example of data input for this model may add clarity and is shown is in Figure 1. This data shows the Supply side input to this model, with P1-P5 being that cadet's Branch preferences. The data in figure 2 shows the Army Branch demand for these cadets. We have 2545 cadets and branch demand also sums to 2545.

OMS_Score	Sex	Race	PB	P1	P2	P3	P4	P5	ac
97.21103	M	2	IN	IN	EN	FA	MP	SC	1
96.859133	F	1	AV	AV	AG	FI	MS	MI	2
96.692956	M	1	AV	AV	MS	AR	MI	IN	2
96.545599	M	1	AV	AV	IN	EN	FA	AR	4
96.221521	M	1	IN	IN	MI	EN	MP	AR	1

Figure 1, Supply: cadet data (5 of 2545) ordered by OMS

PROC NETFLOW takes a data set of nodes and arcs as basic input. Given the data represented in Figures 2 and 3, we should be able to assemble a dataset of nodes (both supply and demand) and the arcs between them. The node data should look like that represented in first column of Figure 4. The supply nodes are positive and the demand nodes are negative, conservation of flow should sum all nodes to zero.

Branch Name	Branch Code	demand
Air Defense	AD	82
Adjutant General	AG	144
Armor	AR	128
Aviation	AV	146
Chemical	CM	67
Engineering	EN	189
Field Artillery	FA	233
Finance	FI	35
Infantry	IN	282
Military Intelligence	MI	362
Military Police	MP	100
Ordinance	OD	208
Quartermaster	QM	164
Signal Corp	SC	255
Transportation Corp	TC	150
	Total	2545

Figure 2, Demand: Branch needs data

After setting up the basic problem, the challenge becomes determining a method for assigning values to arcs (that connect ROTC cadets to assignments). Regular SAS® data step procedures will support this nicely. Starting with all the arcs (cadet to basic branch path, one per branch for every cadet), all arcs are scored as a 1 and the score is subsequently adjusted based on branch preferences and OML scores. This scoring logic is shown below in Figure 3.

- All arcs initially scored as 1
- Score then adjusted for preferences,
  if 1st preference then that arc is given, +5
  if 2nd then +4, if 3rd then +3,
  if 4th then +2, if 5th then +1.
- Score then adjusted for OMS like so,
  new Score = current Score/OMS ranking

Figure 3, Scoring Logic

The Arc Data for cadet 3 is represented in the right side of Figure 4. Note that the arc dataset is assembled in a SAS Data step using logic that builds the arcs from the node data (essentially node data is all from the input files) and the scoring logic mentioned in Figure 3. A graphic of the arcs for cadet 3 are shown in Figure 5. Thicker lines represent the higher scored arcs (here scores are called _cost_, the 'default' name in this procedure).

Node Data (mostly, for cadet 3), supply is positive and demand are negative

node	_sd_
1	1
2	1
3	**1**
4	1
5	1
.	
.	
.	
2544	1
2545	1
ad	-82
ag	-144
ar	-128
av	-146
cm	-67
en	-189
fa	-233
fi	-35
in	-281
mi	-363
mp	-100
od	-208
qm	-164
sc	-255
tc	-150

Arc data for cadet 3 (with scores called _cost_)

Obs	_from_	_cost_	_name_	_to_	_capac_	_lo_
31	3	0.3333	3_ad	ad	1	.
32	3	0.3333	3_ag	ag	1	.
33	3	0.3333	3_cm	cm	1	.
34	3	0.3333	3_en	en	1	.
35	3	0.3333	3_fa	fa	1	.
36	3	0.3333	3_fi	fi	1	.
37	3	0.3333	3_tc	tc	1	.
38	3	0.3333	3_mp	mp	1	.
39	3	0.3333	3_od	Od	1	.
40	3	0.3333	3_qm	Qm	1	.
41	3	0.3333	3_sc	Sc	1	.
42	3	0.6667	3_in	In	1	.
43	3	1	3_mi	mi	1	.
44	3	1.3333	3_ar	ar	1	.
45	3	2	3_av	av	1	.

Figure 4, Node and Arc Data Sets for Cadet 3

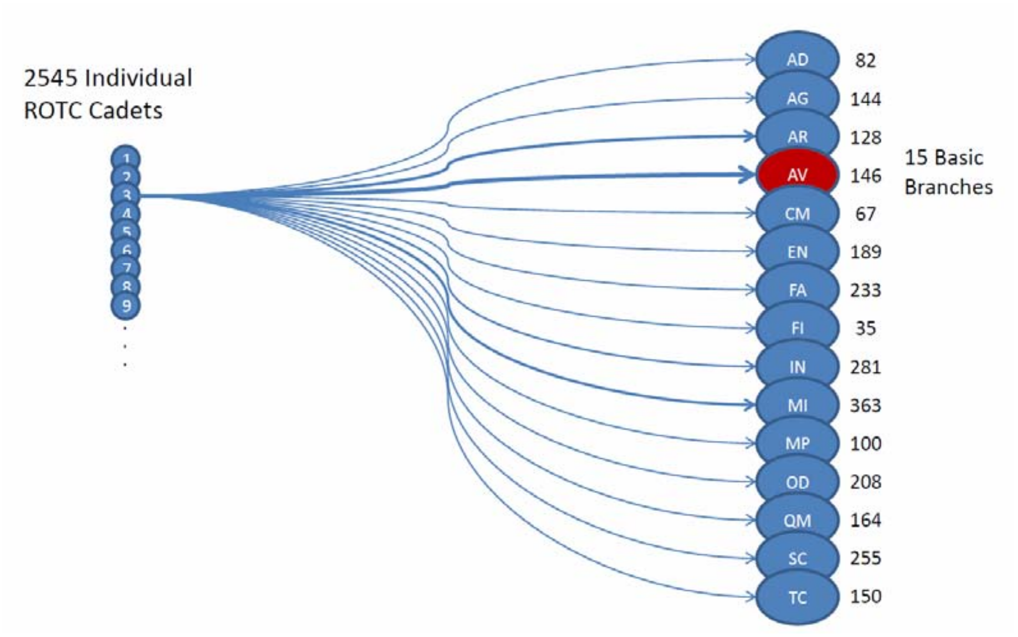

Figure 5, Node and Arc Data for Cadet 3

A standard form representation of this problem is below. The objective function in Figure 6 (below the word maximize), represents 38175 arcs (if written out, this formula would have 38175 variables and coefficients). The constraint equations (below the words 'subject to') represent 2545 possible cadets for each branch.

```
maximize

  1.00*1_ad + 1.00*1_ag + 1.00*1_ar + 1.00*1_av + 1.00*1_cm + 1.00*1_fi +
  1.00*1_mi + 1.00*1_od + 1.00*1_qm + 1.00*1_tc + 2.00*1_sc + 3.00*1_mp +
  4.00*1_fa + 5.00*1_en + 6.00*1_in + 0.50*2_ad + 0.50*2_ar + 0.50*2_cm +
  0.50*2_en + 0.50*2_fa + 0.50*2_in + 0.50*2_mp + 0.50*2_od + 0.50*2_qm +
  0.50*2_sc + 0.50*2_tc + 1.00*2_mi + 2.00*2_fi + 2.50*2_ag + 3.00*2_av +
  0.33*3_ad + 0.33*3_ag
  ....
  0019*2544_ar + .0023*2544_cm + .0003*2545_ag + .0003*2545_av +
  0003*2545_cm + .0003*2545_fa + .0003*2545_in + .0003*2545_mp +
  0003*2545_od + .0003*2545_qm + .0003*2545_sc + .0003*2545_tc +
  0007*2545_mi + .0011*2545_en + .0015*2545_ar + .0019*2545_fi +
  0023*2545_ad

subject too
  1_ad + 2_ad + 3_ad + 4_ad + 5_ad + 6_ad + ... + 2543_ad + 2544_ad + 2545_ad = 82
  1_ag + 2_ag + 3_ag + 4_ag + 5_ag + 6_ag + ... + 2543_ag + 2544_ag + 2545_ag = 144
  1_ar + 2_ar + 3_ar + 4_ar + 5_ar + 6_ar + ... + 2543_ar + 2544_ar + 2545_ar = 128
  1_av + 2_av + 3_av + 4_av + 5_av + 6_av + ... + 2543_av + 2544_av + 2545_av = 146
  1_cm + 2_cm + 3_cm + 4_cm + 5_cm + 6_cm + ... + 2543_cm + 2544_cm + 2545_cm = 67
  1_en + 2_en + 3_en + 4_en + 5_en + 6_en + ... + 2543_en + 2544_en + 2545_en = 189
  1_fa + 2_fa + 3_fa + 4_fa + 5_fa + 6_fa + ... + 2543_fa + 2544_fa + 2545_fa = 233
  1_fi + 2_fi + 3_fi + 4_fi + 5_fi + 6_fi + ... + 2543_fi + 2544_fi + 2545_fi = 35
  1_in + 2_in + 3_in + 4_in + 5_in + 6_in + ... + 2543_in + 2544_in + 2545_in = 282
  1_mi + 2_mi + 3_mi + 4_mi + 5_mi + 6_mi + ... + 2543_mi + 2544_mi + 2545_mi = 362
  1_mp + 2_mp + 3_mp + 4_mp + 5_mp + 6_mp + ... + 2543_mp + 2544_mp + 2545_mp = 100
  1_od + 2_od + 3_od + 4_od + 5_od + 6_od + ... + 2543_od + 2544_od + 2545_od = 208
  1_qm + 2_qm + 3_qm + 4_qm + 5_qm + 6_qm + ... + 2543_qm + 2544_qm + 2545_qm = 164
  1_sc + 2_sc + 3_sc + 4_sc + 5_sc + 6_sc + ... + 2543_sc + 2544_sc + 2545_sc = 255
  1_tc + 2_tc + 3_tc + 4_tc + 5_tc + 6_tc + ... + 2543_tc + 2544_tc + 2545_tc = 150

subject too
   1_ad + 1_ag + 1_ar + 1_av + 1_cm + 1_en + 1_fa + 1_fi + 1_in + 1_mi + 1_mp + 1_ms + 1_od + 1_qm + 1_sc + 1_tc = 1
   2_ad + 2_ag + 2_ar + 2_av + 2_cm + 2_en + 2_fa + 2_fi + 2_in + 2_mi + 2_mp + 2_ms + 2_od + 2_qm + 2_sc + 2_tc = 1
   3_ad + 3_ag + 3_ar + 3_av + 3_cm + 3_en + 3_fa +
   ....
   2545_ad + 2545_ag + 2545_ar + 2545_av + 2545_cm + 2545_en + 2545_fa + 2545_fi + 2545_in + 2545_mi + 2545_mp + 2545_ms +
   2545_od + 2545_qm + 2545_sc + 2545_tc = 1
```

Figure 6, Standard form

The cadet characteristics of branch preference and OML scores are used to assign coefficients for the objective function, and the values in the constraints variables are either 0 or 1.

Now that we have datasets for nodes and arcs, we are all set to run a PROC NETFLOW.

```
proc netflow maximize
  nodedata=node_data
  arcdata=arc_data
  arcout= sol;
  reset maxit1 = 30000;
run;
```

Figure 7, PROC NETFLOW

We also specify an output dataset (arcout) and a maximum number of iterations (reset maxit1 = 30000) as shown in Figure 7.

The log file lines from this NETFLOW procedure are shown below and you can see that an optimum solution was found.

```
NOTE: Number of iterations performed (neglecting any constraints)= 10742 .
NOTE: Of these, 7944 were degenerate.
NOTE: Optimum (neglecting any constraints) found.
NOTE: Maximal total cost= 48.945545982 .
NOTE: The data set WORK38175.SOL has 38175 observations and 14 variables
```

Figure 8, Log file output showing PROC NETFLOW output.

AN INITIAL SOLUTION

Results for cadet 3 (from the sol dataset specified by the arcout clause) show that the cadet has a 'KEY_ARC BASIC' which assigns that cadet to the AV branch.

from	_to_	_cost_	_capac_	_lo_	_name_	_SUPPLY_	_DEMAND_	_FLOW_	_FCOST_	_RCOST_	_ANUMB_	_TNUMB_	_STATUS_	
3	ad	0.33333	1	0	3_ad	1	82	0	0	-1.66	11836	33	LOWERBD	NONBASIC
3	ag	0.33333	1	0	3_ag	1	144	0	0	-1.66	13594	33	LOWERBD	NONBASIC
3	ar	1	1	0	3_ar	1	128	0	0	-1	34688	33	LOWERBD	NONBASIC
3	av	2	1	0	3_av	1	146	1	2	.	36445	33	KEY_ARC	BASIC
3	cm	0.33333	1	0	3_cm	1	67	0	0	-1.66	15352	33	LOWERBD	NONBASIC
3	en	0.33333	1	0	3_en	1	189	0	0	-1.66	17110	33	LOWERBD	NONBASIC
3	fa	0.33333	1	0	3_fa	1	233	0	0	-1.66	18868	33	LOWERBD	NONBASIC
3	fi	0.33333	1	0	3_fi	1	35	0	0	-1.66	20625	33	LOWERBD	NONBASIC
3	in	0.33333	1	0	3_in	1	281	0	0	-1.67	22383	33	LOWERBD	NONBASIC
3	mi	0.66667	1	0	3_mi	1	363	0	0	-1.33	32930	33	LOWERBD	NONBASIC
3	mp	0.33333	1	0	3_mp	1	100	0	0	-1.66	24140	33	LOWERBD	NONBASIC
3	od	0.33333	1	0	3_od	1	208	0	0	-1.66	25898	33	LOWERBD	NONBASIC
3	qm	0.33333	1	0	3_qm	1	164	0	0	-1.66	27656	33	LOWERBD	NONBASIC
3	sc	0.33333	1	0	3_sc	1	255	0	0	-1.66	29414	33	LOWERBD	NONBASIC
3	tc	0.33333	1	0	3_tc	1	150	0	0	-1.66	31172	33	LOWERBD	NONBASIC

Figure 9, arc data from sol dataset
(_cost_ column is what we are maximizing)

Assignment results, shown below in Figure 10, are for the first and last cadets (highest and lowest OMS). You can see that the _cost_ variable is used to maximize cadet satisfaction. The more cadets get their 1st choice, the larger the aggregate solution. This solution has a maximum **cost = 48.945545982**, from the log file output of Figure 8.

RANK	SEX	OMS	cost	assigned	BR1	BR2	BR3	BR4	BR5	choice
1	M	97.2110	6	IN	IN	EN	FA	MP	SC	1st
2	F	96.8591	3	AV	AV	AG	FI	MS	MI	1st
3	M	96.6930	2	AV	AV	MS	AR	MI	IN	1st
4	M	96.5456	1.5	AV	AV	IN	EN	FA	AR	1st
5	M	96.2215	1.2	IN	IN	MI	EN	MP	AR	1st
6	M	95.9808	1	EN	EN	MI	IN	AR	MS	1st
7	M	95.5532	0.85714286	MP	MP	EN	MI	AR	FA	1st
				.						
				.						
				.						
2536	M	65.3007	0.00039432	OD	IN	AR	EN	MI	FA	other
2537	M	64.9649	0.002365	TC	TC	AD	FA	SC	MI	1st
2538	M	64.5997	0.00236407	AD	AD	AR	MI	FA	IN	1st
2539	M	64.5783	0.00236314	QM	QM	SC	AR	TC	EN	1st
2540	M	64.3637	0.0023622	SC	SC	MI	FA	MP	OD	1st
2541	M	64.1470	0.00196773	FA	MP	FA	AD	EN	CM	2nd
2542	M	63.8208	0.00236035	QM	QM	SC	OD	FA	AR	1st
2543	M	63.6510	0.00235942	TC	TC	FA	QM	AD	MP	1st
2544	M	63.1117	0.00235849	CM	CM	AR	IN	MI	MP	1st
2545	M	59.7413	0.00235756	AD	AD	FI	AR	EN	MI	1st

Figure 10, Cadet Assignments compared to cadet preference.

The Demand objectives are filled by this solution and Cadet Satisfaction is maximized as shown in Figure 11 below.

Demand (Goal)				Cadet Satisfaction (1st, 2nd, or 3rd preference matched)			
BR	ASSIGNED	PERCENT	GOAL				
AD	82	3.222	82				
AG	144	5.6582	144				
AR	128	5.0295	128				
AV	146	5.7367	146	choice	COUNT	PERCENT	CUM
CM	67	2.6326	67	1st	1572	61.768	61.768
EN	189	7.4263	189	2nd	640	25.147	86.916
FA	233	9.1552	233	3rd	253	9.941	**96.857**
FI	35	1.3752	35	4th	41	1.611	98.468
IN	281	11.0413	281	5th	14	0.550	99.018
MI	363	14.2633	363	other	25	0.982	100.000
MP	100	3.9293	100				
OD	208	8.1729	208				
QM	164	6.444	164				
SC	255	10.0196	255				
TC	150	5.8939	150				

Figure 11, Assignment results summary

96.857% cadet satisfaction is pretty good, especially when we are exactly matching the Army branch need (Demand), but we have ignored the additional constraints regarding gender and combat arms, and quality.

The constraint that prohibits females from being assigned to the Combat Arms (IN, FA, and AR) branches can be implemented in NETFLOW Procedure by simply deleting the arcs between female cadets and those three branches from the arcs dataset. What happens if you wanted to add constraints to proportionally distribute females to the remaining branches? You can see from Figure 12 below that the distribution of cadets by 1st preference when compared to the distribution of cadets after the demand constraint is applied. The 1st preference distribution (on the left) makes the AG branch almost 50% female. A proportional distribution by gender constraint could be used to make all permitted branches more representative of the whole population.

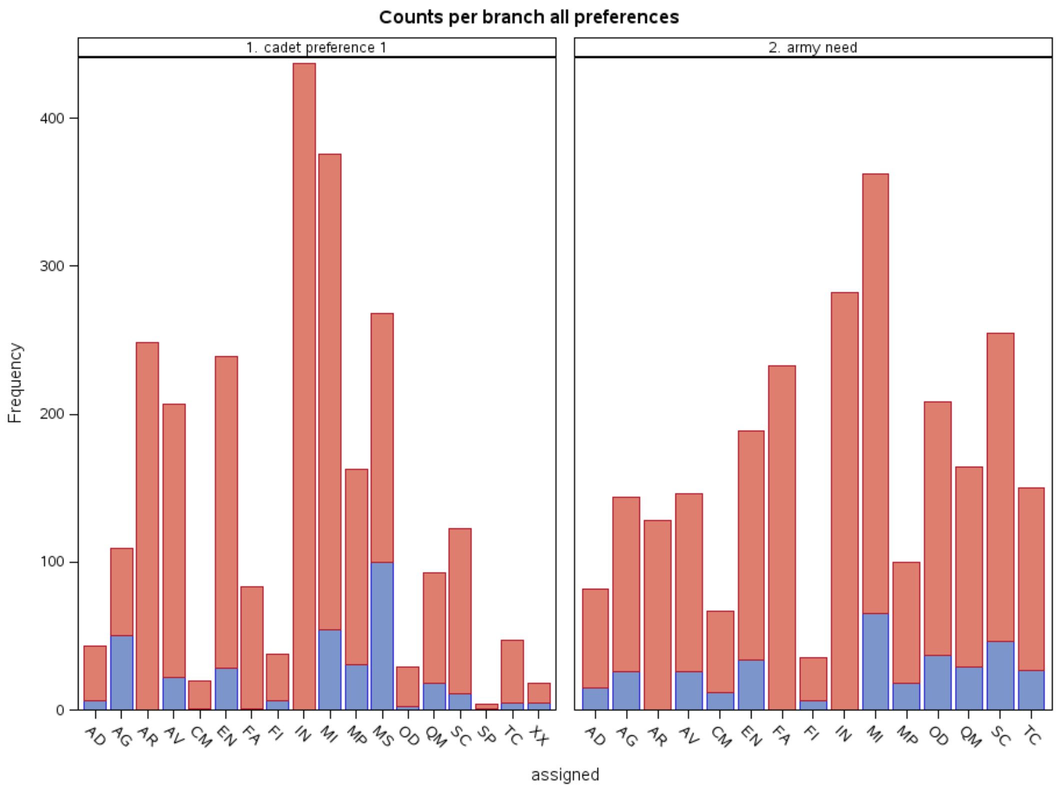

Figure 12, Cadet Preference compared to cadet assignment.

The NETFLOW procedure supports applying proportional distribution constraints (via ‘intermediate nodes’) and subsequently applying demand side constraints (‘side constraints’), but the OPTMODEL syntax better supports the application of many proportional distribution and demand side constraints.

AN OPTMODEL SOLUTION

An OPTMODEL version of this network model will be developed before we add any more constraints. The OPTMODEL procedure adds language to support the Operations Research (OR) Math Programming Language style of modeling using index sets and associated logic constructs. The data sets of nodes from the prior model will be immediately read into Index Sets, we’ll create an arcs array and we’ll re-run the model above. An example of running input data into a data set, then an index set is shown below. To create the cadets and goals data sets, read in the cadets.csv file and the goals.txt file

```
filename cadet './ins/cadets.csv';
filename goal './ins/goal.txt';
/* read in the cadet data */
data cadets_loaded;
  infile cadet DLM=',' DSD MISSOVER;
  input OMS $ SEX $ RACE $ prevbr $ BR1 $ BR2 $ BR3 $ BR4 $ BR5 $ acd $ ;
  if _N_ > 1;
  omsnbr = input(OMS,9.);
  ac = input(acd,9.);
  RCE='1';
  if RACE ne '1' then RCE='2';
run;

proc sort data=cadets_loaded;
   by descending omsnbr;

data cadets;
  set cadets_loaded;
  rank=_N_;
run;

/* this reads in the demand data goals.txt */
data goalst(drop=goal);
  infile goal missover;
  input BR $1-2 goal $ ;
  goalnbr = input(goal,9.);
run;
```

Then do some sql and data stepping to expand the goals dataset to support additional constraints and some calculations.

```
proc sql; select count(*) as cnt into :all_cnt from cadets; quit;
proc sql; select count(*) into :fem_cnt from cadets where SEX='F'; quit;
proc sql; select count(*) into :a1_cnt from cadets where ac = 1 ; quit;
proc sql; select count(*) into :a2_cnt from cadets where ac = 2 ; quit;
proc sql; select count(*) into :a3_cnt from cadets where ac = 3 ; quit;
proc sql; select count(*) into :a4_cnt from cadets where ac = 4 ; quit;
proc sql; select count(*) into :min_cnt from cadets where RACE ne '1' ; quit;
proc sql; select count(*) into :min2_cnt from cadets where RACE eq '2' ; quit;
proc sql; select count(*) into :min3_cnt from cadets where RACE eq '3' ; quit;
proc sql; select count(*) into :min4_cnt from cadets where RACE eq '4' ; quit;
proc sql; select count(*) into :min5_cnt from cadets where RACE eq '5' ; quit;
proc sql; select avg(omsnbr) into :avg_oms from cadets; quit;
proc sql; select sum(goalnbr) as cnt into :ca_cnt from goalst where BR in
('IN','AR','FA'); quit;

data goals;
  set goalst;
  femgoals=round(goalnbr * &fem_cnt/(&all_cnt-&ca_cnt),1);
  if BR in ('IN' 'AR' 'FA') then femgoals = 0;
  mingoals=round(goalnbr * &min_cnt/&all_cnt,1);
```

```
    min2goals=round(goalnbr * &min2_cnt/&all_cnt,1);
    min3goals=round(goalnbr * &min3_cnt/&all_cnt,1);
    min4goals=round(goalnbr * &min4_cnt/&all_cnt,1);
    min5goals=round(goalnbr * &min5_cnt/&all_cnt,1);
    a1goals=round(goalnbr * &a1_cnt/&all_cnt,1);
    a2goals=round(goalnbr * &a2_cnt/&all_cnt,1);
    a3goals=round(goalnbr * &a3_cnt/&all_cnt,1);
    a4goals=round(goalnbr * &a4_cnt/&all_cnt,1);
    qgoals=0.998*goalnbr*&avg_oms;
  run;
```

Call the OPTMODEL procedure and read the cadets and the goals data sets into index sets (load 'nodes' in network terms), create an arcs index set, and declare an index set of decision variables x. You can check the index set contents, by looking into your output file for the results of the print statements.

```
proc optmodel printlevel=2;

  set C;
  string OMS{C};
  string SEX{C};
  string RACE{C};
  string prevbr{C};
  string BR1{C};
  string BR2{C};
  string BR3{C};
  string BR4{C};
  string BR5{C};
  number omsnbr{C};
  number ac{C};
  string RCE{C};
  number rank{C};
  read data cadets into C=[_N_]OMS SEX RACE prevbr
             BR1 BR2 BR3 BR4 BR5 omsnbr ac RCE rank;

  print OMS SEX RACE prevbr BR1 BR2 BR3 BR4 BR5 omsnbr ac RCE rank;

  set B;
  string BR{B};
  number goalnbr{B};
  number femgoals{B};
  number mingoals{B};
  number min2goals{B};
  number min3goals{B};
  number min4goals{B};
  number min5goals{B};
  number a1goals{B};
  number a2goals{B};
  number a3goals{B};
  number a4goals{B};
  number qgoals{B};
  read data goals into B=[_N_] BR goalnbr femgoals mingoals
            min4goals min3goals min2goals min5goals a1goals
            a2goals a3goals a4goals qgoals;

  print BR goalnbr femgoals mingoals min4goals min3goals min2goals
            min5goals a1goals a2goals a3goals a4goals qgoals;
```

Create the arcs index set and declare an x decision variable.

```
  number arc{B,C};

  for{i in B, j in C}
  if BR[i]=BR1[j] then arc[i,j] = 5/rank[j];
```

```
     else if BR[i]=BR2[j] then arc[i,j] = 4/rank[j];
     else if BR[i]=BR3[j] then arc[i,j] = 3/rank[j];
     else if BR[i]=BR4[j] then arc[i,j] = 2/rank[j];
     else if BR[i]=BR5[j] then arc[i,j] = 1/rank[j];
     else arc[i,j]=0;
     /*  print arc;  */
```

Create subsets of the cadets index set, like all females, or academic degree type 1, or all redcat categories. Generally any attribute of the cadet population that you may want to proportionally distribute, can be taken into a subset here with the setof function supported by OPTMODEL.

```
     set CA = setof{i in B: BR[i] eq 'IN' OR BR[i] eq 'AR' OR BR[i] eq 'FA'}<i>;
     set NCA = setof{i in B: BR[i] ne 'IN' AND BR[i] ne 'AR' AND BR[i] ne 'FA'}<i>;
       print{i in CA} BR[i];

     print{i in NCA} BR[i];

     var x{B,C} >= 0;
```

Using the decision variables in the x index set and the scores in the arcs index set create a maximization equation.

```
        maximize total_score = sum{i in B, j in C}arc[i,j]*x[i,j];
        constraint supply{j in C}: sum{i in B}x[i,j]=1;
        constraint demand{i in B}: sum{j in C}x[i,j]>=goalnbr[i];
        constraint fems_no_ca{i in CA}: sum{j in F}x[i,j]=0;

      /*constraint fcadem{i in B}: sum{j in F}x[i,j]>=femgoals[i];
        constraint quality{i in B}: sum{j in C}omsnbr[j]*x[i,j]>=qgoals[i];
        constraint mindem{i in B}: sum{j in M}x[i,j]>=mingoals[i];
        constraint min2dem{i in B}: sum{j in R2}x[i,j]>=min2goals[i];
        constraint min3dem{i in B}: sum{j in R3}x[i,j]>=min3goals[i];
        constraint min4dem{i in B}: sum{j in R4}x[i,j]>=min4goals[i];
        constraint min5dem{i in B}: sum{j in R5}x[i,j]>=min5goals[i];
        constraint a1type{i in B}: sum{j in A1}x[i,j]>=a1goals[i];
        constraint a2type{i in B}: sum{j in A2}x[i,j]>=a2goals[i];
        constraint a3type{i in B}: sum{j in A3}x[i,j]>=a3goals[i];
        constraint a4type{i in B}: sum{j in A4}x[i,j]>=a4goals[i];*/
        solve;
```

Take the x set of decision variables that are greater then zero and place the assignment, rank (here cadet number), arc scores, etc… and place those values in a data set called solt.

```
        /* print x;  */
        create data solt from [B C]={i in B, j in C: x[i,j]>0} BR[i] rank[j] arc[i,j]
               ac[j] SEX[j] RACE[j] RCE[j] prevbr[j] BR1[j] BR2[j] BR3[j]
               BR4[j] BR5[j] omsnbr[j];
      quit;
```

The data set solt (temporary solution) is now available to support analysis and expansion. The code below categorizes the result and assigns a random number to each assignment to support disambiguation of results.

```
      data sol;
        set solt(rename=(BR=assigned arc=score));
        choice = '6other';
        if assigned = br1 then choice = '1st';
        else if assigned = br2 then choice = '2nd';
        else if assigned = br3 then choice = '3rd';
        else if assigned = br4 then choice = '4th';
        else if assigned = br5 then choice = '5th';
        _random = RANUNI(0);
        rnd = put(_random,10.8);
        rnk = put(rank,6.);
```

```
    obs = put(_N_,6.);
    scorec=put(score,10.8);
  run;
```

Some output from this data set is shown below in Figure 13 and you can see that the top ranked cadets are generally getting their 1st choice assigned to them, demand is met, and satisfaction is 96.817%.

Assignments Listing, the first 10 rows

B	C	assigned	rank	score	ac	SEX	OMS	BR1	BR2	BR3	BR4	BR5	choice	Rnd
9	1	IN	1	5	1	M	97.211	IN	EN	FA	MP	SC	1st	0.746979
4	2	AV	2	2.5	2	F	96.8591	AV	AG	FI	MS	MI	1st	0.749089
4	3	AV	3	1.66667	2	M	96.693	AV	MS	AR	MI	IN	1st	0.776854
4	4	AV	4	1.25	4	M	96.5456	AV	IN	EN	FA	AR	1st	0.691201
9	5	IN	5	1	1	M	96.2215	IN	MI	EN	MP	AR	1st	0.729527
6	6	EN	6	0.83333	2	M	95.9808	EN	MI	IN	AR	MS	1st	0.748766
12	7	MP	7	0.71429	1	M	95.5532	MP	EN	MI	AR	FA	1st	0.87659
9	8	IN	8	0.625	4	M	95.2922	IN	AR	EN	MI	OD	1st	0.979897
6	9	EN	9	0.55556	1	M	94.8416	EN	FI	MI	SC	AR	1st	0.861697
10	10	MI	10	0.5	1	F	94.7545	MI	MS	EN	FI	AD	1st	0.724443

Demand (Goal)

BR	ASSIGNED	PERCENT	GOAL
AD	82	3.222	82
AG	144	5.6582	144
AR	128	5.0295	128
AV	146	5.7367	146
CM	67	2.6326	67
EN	189	7.4263	189
FA	233	9.1552	233
FI	35	1.3752	35
IN	281	11.0413	281
MI	363	14.2633	363
MP	100	3.9293	100
OD	208	8.1729	208
QM	164	6.444	164
SC	255	10.0196	255
TC	150	5.8939	150

Cadet Satisfaction (1st, 2nd, or 3rd preference matched)

CHOICE	COUNT	SATISFIED	CUMULATIVE
1st	1569	61.6503	61.65
2nd	644	25.3045	86.955
3rd	251	9.8625	96.817
4th	41	1.611	98.428
5th	14	0.5501	98.978
6other	26	1.0216	100

Figure 13, Assignments summary from PROC OPTMODEL

Now let's add in some more constraints. An example of a proportional distribution constraint is shown in the code above (page 9) and below. To enable the female proportional distribution constraint in the problem above, we uncomment this constraint and re-run the model.

```
        constraint fcadem{i in B}: sum{j in F}x[i,j]>=femgoals[i];
```

Note that the FGOAL numbers are calculated by the counting sql statements in the first section of the code above where we loaded the text files into datasets and extended those datasets. The specific lines that calculated femgoals are shown below.

```
proc sql; select count(*) as cnt into :all_cnt from cadets; quit;
proc sql; select count(*) into :fem_cnt from cadets where SEX='F'; quit;
proc sql; select sum(goalnbr) as cnt into :ca_cnt from goalst where BR in
('IN','AR','FA'); quit;

data goals;
  set goalst;
  femgoals=round(goalnbr * &fem_cnt/(&all_cnt-&ca_cnt),1);
  if BR in ('IN' 'AR' 'FA') then femgoals = 0; …
run;
```

Some output from this run is shown below in Figure 14 and you can see effects of adding the gender proportional distribution constraint, demand is met, satisfaction is 95.678%.

Female distribution before constraint					Female distribution After constraint					Satisfaction	
BBR	ASGN	FEMS	FGOAL	DELTA	BBR	ASGN	FEMS	FGOAL	DELTA	CHOICE	CUMULATIVE
AD	82	15	15	0	AD	82	15	15	0		
AG	144	69	26	43	AG	144	26	26	0		
AR	128	0	0	0	AR	128	0	0	0		
AV	146	15	26	-11	AV	146	26	26	0		
CM	67	10	12	-2	CM	67	12	12	0		
EN	189	23	34	-11	EN	189	34	34	0	1st	60.629
FA	233	0	0	0	FA	233	0	0	0	2nd	85.383
FI	35	4	6	-2	FI	35	6	6	0	3rd	95.678
IN	282	0	0	0	IN	282	0	0	0	4th	97.957
MI	362	62	65	-3	MI	362	65	65	0	5th	98.546
MP	100	22	18	4	MP	100	18	18	0	6other	100
MS	0	0	0	0	MS	0	0	0	0		
OD	208	25	37	-12	OD	208	37	37	0		
QM	164	43	29	14	QM	164	29	29	0		
SC	255	27	46	-19	SC	255	46	46	0		
TC	150	26	27	-1	TC	150	27	27	0		

Figure 14, Distribution of females and effects on satisfaction.

An example of a demand side constraint (called a 'side constraint' in NETFLOW) is demonstrated here using the OMS scores of each candidate to calculate the average OMS score of each branch (demand side) after the assignments are made. The objective of such a constraint will be to make the average OMS scores of each branch assignment more nearly equal so that one branch does not get a disproportionate share low or high scoring cadets.

The code for the proc sql 'counting' and calculations to do support this demand side constraint is shown below. Note that the problem will fail if you demand that the average OMS score per branch is exactly equal to the overall cadet sample average, so we need to 'factor back from 1' to get a slightly lower quality goal per branch.

```
proc sql; select avg(omsnbr) into :avg_oms from cadets; quit;

data goals;
  set goalst;
   qgoals=0.998*goalnbr*&avg_oms;   …
run;
```

The 'side constraint' simply sums the assigned cadets OMS scores to satisfy the constraint below.

```
constraint quality{i in B}: sum{j in C}omsnbr[j]*x[i,j]>=qgoals[i];
```

The result of re-running the problem with these constraints is shown below in figure 15. You can see from figures 14 and 15 that as you add in constraints to shape the assignments with the additional constraints of proportionally distributing females or more evenly distributing OMS scores among the branches, our ability to satisfy cadets and meet the demand constraint is degraded (about 1.3% for the female proportional distribution and about 5% for the quality constraint). Additionally, for the quality 'side constraint' our ability to meet the demand is degraded, but the SAS facility supports any additional logic necessary to disambiguate fractional assignments (an exercise left for the reader, but fully supported by the details in this document (see the random number added to the sol data set earlier)).

Quality distribution before constraint	Quality distribution After constraint	Satisfaction	Demand

ASSIGNED	AVGOMS
IN	85.69
AV	85.57
AR	83.32
MI	82.03
FI	81.52
EN	81.07
MP	80.85
AG	79.00
SC	77.59
AD	76.87
QM	76.85
FA	76.25
TC	74.84
CM	74.71
OD	74.69

ASSIGNED	AVGOMS
IN	80.60
AV	80.43
OD	79.58
MI	79.58
AG	79.57
AR	79.57
SC	79.57
FI	79.54
TC	79.53
MP	79.53
FA	79.53
CM	79.52
QM	79.52
EN	79.52
AD	79.48

CHOICE	CUMULATIVE
1st	54.892
2nd	78.389
3rd	90.295
4th	93.281
5th	94.931
6other	100

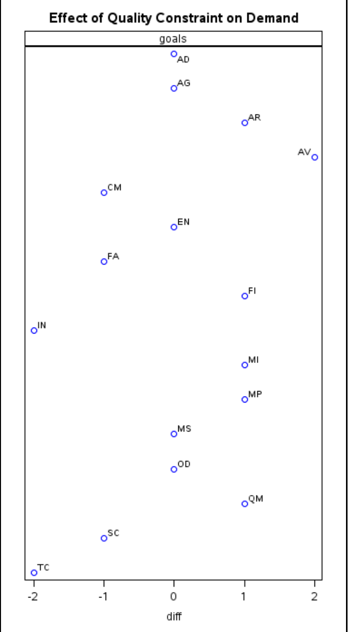

Figure 15, Distribution of Quality and effects on satisfaction.

CONCLUSION

This paper described the business problem of matching cadets to their assignment preferences as constrained by available assignments and other Army needs. Then the paper described a generalized network optimization approach (via nodes and arcs) to solving this problem initially using the NETFLOW procedure, but evolving that same problem to the OPTMODEL procedure showing the effects of a supply side constraint (proportional distribution based on gender) and a demand side constraint (quality distribution). The SAS® facilities ability to take data into data sets from almost any data source, then via the OPTMODEL procedure, feed them to the Operations Research (OR) index sets form for optimization (the OPTMODEL syntax appears to be based on AMPL[1] or GMPL[2]), and back out of OPTMODEL into data sets for results analysis is a capability unmatched in the OR community.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mike Grierson
mcg@mcg-ct.com
www.mcg-ct.com

see more about this presentation at http://www.sascommunity.org/
A_Network_Optimization_Solution_Using_SAS/OR_Tools_for_the_Department_of_the_Army_Branching_Problem

[1] Robert Fourer, David M. Gay and Brian W. Kernighan, "A Modeling Language for Mathematical Programming", *Management Science* 36 (1990) 519-554.
[2] Andrew Makhorin, Modeling Language GNU MathProg Language Reference Draft Edition, for GLPK Version 4.42, January 2010, http://www.gnu.org/software/glpk/