Paper 141-2008

Assigning ROTC Cadets into Air Force Specialty Codes Using SAS OR's PROC LP Procedure

Anthony Lawrence, Air Force Personnel Center (AFPC)/Assignment Programs and Procedures Division, Randolph AFB, TX 78150

ABSTRACT

SAS OR is used to solve a variety of Optimization Problems. This presentation describes how SAS OR's Proc LP procedure with the help of SAS' Proc SQL procedure is used to assign ROTC cadets to an occupational field (Air Force Specialty Code).

INTRODUCTION

In order to solve a linear program (LP) either by hand or using a programming language such as SAS OR's Proc LP procedure, we must first be able to formulate the problem. A LP problem has three parts: the decision variables, the constraints, and the objective function. The decision variables are variables whose quantities we are trying to optimize. The constraints are limits on the values that the decision variables can become. The objective function is a linear function of the decision variables that is maximized, minimized, or balanced.

Real world application: Every year the Air Force must bring graduating ROTC cadets on to Active Duty service. Last year approximately 1350 non-flying ROTC cadets were brought on Active Duty. Prior to entering Active Duty, the Air Force Personnel Center assigns ROTC cadets to an occupational field (known as Air Force Specialty Code, or AFSC).

A BASIC OPTIMIZATION PROBLEM:

To better understand the ROTC optimization problem we will go over a simple example that mimics the ROTC optimization problem.

Graduation and Occupation example

Given:

You have four future graduates and you must assign each to an occupation. There are two occupations that each student can be assigned to. We are assuming that all four students are qualified for the two occupations. Each student desires a certain occupation more than the other. A student can only be assigned to one occupation. We must maximize the sum of desired levels to make each graduate happy with their new occupations.

4 students => Andrew, Brian, Chris, and Dave
2 occupations => Engineer and Mechanic

There can only be 2 engineers and 2 mechanics

Occupation desire levels: 5 being their dream job and 1 not wanting the occupation

Formulation:

Decision variables:	Desire Level:
AE = Andrew becoming an engineer	3
AM = Andrew becoming a mechanic	2
BE = Brian becoming an engineer	4

```
BM = Brian becoming a mechanic          2
CE = Chris becoming an engineer         3
CM = Chris becoming a mechanic          3
DE = Dave becoming an engineer          5
DM = Dave becoming a mechanic           1

Objective Function:
Maximize 3AE + 2AM + 4BE + 2BM + 3CE + 3CM + 5DE + 1DM

Constraints:
Each student can only be assigned to one occupation:
AE + AM = 1 (Andrew)
BE + BM = 1 (Brian)
CE + CM = 1 (Chris)
DE + DM = 1 (Dave)

There can only be engineers:
AE + BE + CE + DE <= 2

There can only be mechanics:
AM + BM + CM + DM <= 2

Non-negativity:
All decision variables must be greater than or equal to 0
(If not specified PROC LP assumes that the lower bound for a decision
 variable is zero).
```

Putting the problem into SAS:

The most common format for PROC LP data is the sparse format. The sparse format enables you to enter only the nonzero coefficients into a SAS dataset. The SAS datasets that describes a sparse model must contain a type variable, a column variable, a row variable and a coefficient variable. The type variable tells how PROC LP interprets each row in the dataset formulation. If the row is the objective function then the type variable can be MAX or MIN. If the row is a constraint variable then the type variable can be LE (less than or equal to), GE, EQ, UPPERBD, LOWERBD and other math functions. The column variables are used to define the decision variables and to identify which coefficients are the right hand side values for the constraints. The row variables are the names of the rows that make up the objective function and the constraints. The coefficient variables are either values associated with each decision variable or the values on the right hand side of a constraint. SAS OR's Proc LP procedure automatically recognizes the names _TYPE_, _COEF_, _COL_, and _ROW_ as the type, coefficient, column, and row variables, respectively.

Translating our mathematical formulation into a SAS dataset:

```
DATA DATA1;
INPUT
_TYPE_ $    _COEF_     _COL_ $      _ROW_ $;

DATALINES;
MAX             3          AE           OBJ
.               2          AM           OBJ
.               4          BE           OBJ
.               2          BM           OBJ
.               3          CE           OBJ
.               3          CM           OBJ
```

Understanding the code:
The first 8 lines describe the objective function. If we look at the _ROW_ column the first 8 lines are OBJ. This indicates that the first 8 lines belong to the objective function. Reading across we get:

3AE + 2AM + 4 BE……if we keep reading across the line we will get the mathematical formulation we started with.

```
.               5           DE          OBJ
.               1           DM          OBJ
EQ              1           AE          ANDREW
.               1           AM          ANDREW
.               1           _RHS_       ANDREW
EQ              1           BE          BRIAN
.               1           BM          BRIAN
.               1           _RHS_       BRIAN
EQ              1           CE          CHRIS
.               1           CM          CHRIS
.               1           _RHS_       CHRIS
EQ              1           DE          DAVE
.               1           DM          DAVE
.               1           _RHS_       DAVE
LE              1           AE          ENGINEER
.               1           BE          ENGINEER
.               1           CE          ENGINEER
.               1           DE          ENGINEER
.               2           _RHS_       ENGINEER
LE              1           AM          MECHANIC
.               1           BM          MECHANIC
.               1           CM          MECHANIC
.               1           DM          MECHANIC
.               2           _RHS_       MECHANIC
;
RUN;
```

Solving the optimization problem:
After the sparse data is built and entered into a SAS dataset, we run SAS's linear programming procedure PROC LP. These lines of SAS code below will solve our problem and will provide us with an optimized solution. It will also provide problem summary, solution summary, variable summary, and constraint summary to use in an analysis of the problem.

```
PROC LP SPARSEDATA;
    TYPE _TYPE_;
    ROW _ROW_;
    COL _COL_;
    COEF _COEF_;
ODS OUTPUT VARIABLESUMMARY = RESULTS;
RUN;
```

THE ROTC CLASSIFICATION OPTIMIZATION MODEL:

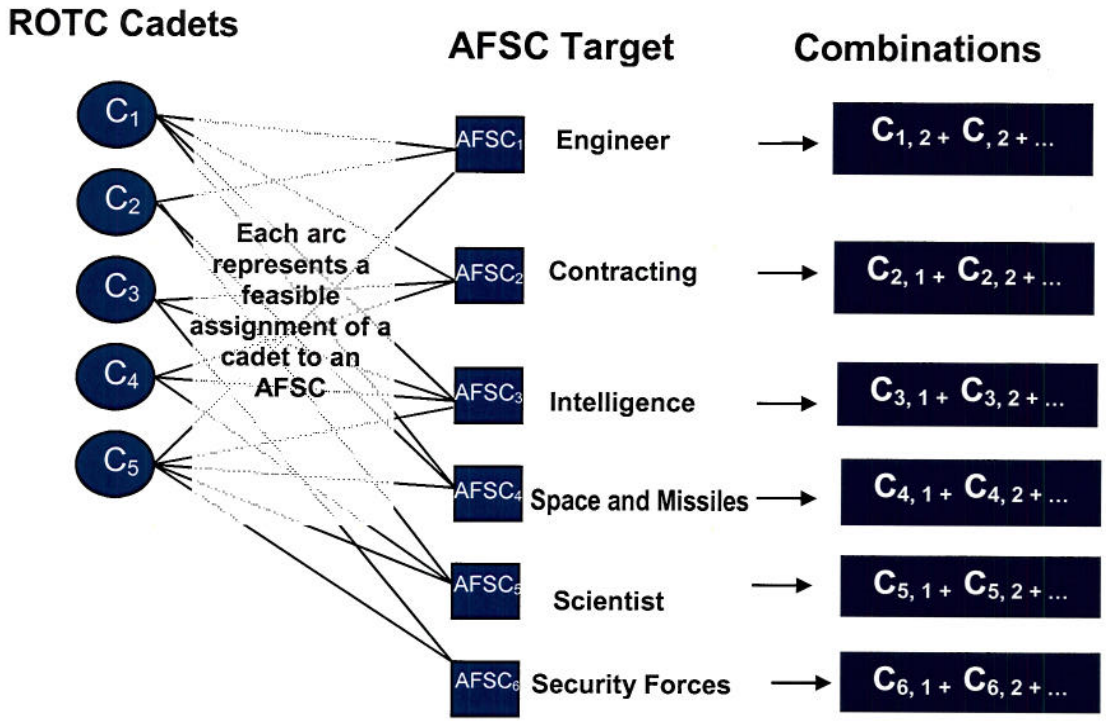
Understanding the ROTC classification model:
The ROTC classification model assigns ROTC cadets into an AFSC. If we compare the ROTC classification model to the simple example above, the graduates are now the ROTC cadets and the occupations are now the AFSCs.

The ROTC classification:
Using the data from 2007 ROTC classification, we will show how multiple datasets are created and combined to create the dataset for the sparse data. Every year the Air Force assigns ROTC cadets to an AFSC. Last year about 1350 ROTC cadets were assigned their AFSC. Each cadet can choose up to six AFSCs that their academic degree will allow them to be qualified for. There are a total of 33 different AFSCs. The input data are the accession targets

(required min/max number of cadets for each AFSC), cadets' AFSC preferences and academic information, Order of Merit from HQ AFROTC (a listing of the cadets, ranked 1 through n), and degree mapping and qualification data from AFPC. After the merging and scrubbing of the input data, we end up with two major datasets which will be used to create the datasets for the sparse data. The two datasets are Cadets and Targets. The Cadets datasets contains information about the cadets, such as order of merit, their six AFSC choices, language skills, and type of degree. The targets dataset contains all 33 AFSCs and number of officers needed for each AFSC.

Objective Function:
The objective to this problem is to maximize the cadets' preference value (cadet's desire for an occupation) and the Air Force's needs for a certain AFSC (Air Force's desires) for assigning ROTC cadets to an AFSC.

A Graphical Representation of possible cadet and AFSC combinations (i.e. Decision Variables):



Decision Variables:
A cadet is not only matched to the six AFSCs they chose, but also to every AFSC for which he/she is qualified based on academic education. When only considering the simplified graphic above, at a minimum, a cadet can be assigned to one of six possible AFSCs. For each cadet there are six lines coming out and each line points to six AFSCs. Using last year's data, 1350 cadets and 6 choices per cadet the basic problem results in 8100 (=1350*6) decision variables of cadet and AFSC combinations. In reality there will be more than 8100 decision variables because in addition to the cadet's six choices we will also have decision variables from every AFSC that a cadet qualifies for based on their academic degree.

Constraints:

As with the previous example, a cadet can only be assigned to one AFSC and for each AFSC we have a minimum and maximum number of requirements.

Formulation:

Decision variables:

$C_{i,j}$ = Cadet/AFSC combinations
(Where i represents cadets 1 to 1355, j represents AFSCs 1 to 33)

The coefficients for each Cadet/AFSC combination come from the weighting factors specified by HQ AFPC. The cadet's degree determines AFSC qualification.

Factors weighted are:
Air Force's needs
Cadets order of merit (academic ranking)
Cadets Preference (6 choices)
Language Skills for the Intelligence career field
Cadets ranking of his/her AFSC choices (importance to cadet)

Letters A through Z represents the value of the weight for each cadet/AFSC combination.

Objective Function:

Max Z = $\sum AC_{1,1} + .. FC_{1,6} + AC_{2,1} + .. FC_{2,6} + AC_{3,1} + .. FC_{3,6} + \longrightarrow AC_{1355,1} + .. FC_{1355,6}$
(All 1355 cadets and their choices)

Constraints:

$\sum C_{1,j} = 1$ (Cadet1)

$\sum C_{2,j} = 1$ (Cadet2)

$\sum C_{3,j} = 1$ (Cadet2)

.
.
.

$\sum C_{1300,j} = 1$ (Cadet1300)

Each row represents all possible cadet/AFSC combinations for a single cadet. A cadet can only be assigned to only one AFSC

. . . . . .

$\sum$ Upper $C_{i,1}$ <= X (AFSC1)

$\sum$ Lower $C_{i,1}$ >= Y (AFSC1)

.
.

$\sum$ Upper $C_{i,33}$ <= X (AFSC33)

$\sum$ Lower $C_{i,33}$ >= Y (AFSC33)

Each row represents all possible cadet/AFSC combinations for a single AFSC. Each AFSC has a number of slots they must fill and limits on how many cadets can go into that AFSC. In mathematical terms, we have a lower and upper bound for each AFSC.

Non-negativity Constraints: If not specified PROC LP assumes that the lower bound for a decision variable is zero.
Integer/Binary: All the decision variables can only take on a value of 0 or 1

Putting the Problem into SAS:
On the simple example provided, we created the sparse data by typing in the values for the Proc LP variables type, column, row, and coefficient. The example only had 8 decision variables and 7 constraints. When we typed in the data for the example we had around 40 lines of SAS code. Creating the sparse data for ROTC classification model by hand would be very time consuming, with more than 8100 decision variables and over 5000 constraints. Translating decision variables and constraints to SAS, means that you will have 8100+ lines of code for the constraints that restrict cadets to only one AFSC. In addition, you will also have 8100+ total lines of code for all 33 AFSCs, for the constraints that restrict the number of cadets assigned to each AFSC; and you will have 8100+ lines for each integer and bound contraints. In other words, you would have to type in over 100,000 lines of code to create your sparse data.

One way to avoid entering all the data by hand is to create a matrix of the datasets cadets and targets. This is where another SAS procedure, Proc SQL, will help us. Proc SQL will match every cadet to the 33 AFSCs. In other words, the table created by the matrix will have over 44550 (=1350*33) rows.

Creating the Matrix:

```
PROC SQL;
CREATE TABLE A1 AS
SELECT *
FROM Cadets, Targets;
QUIT;
```

Now that we have the matrix we must check every AFSC match with each cadet to see if the cadet's academic degree qualifies them for that AFSC. If the degree does not qualify a cadet for an AFSC, then that row will be deleted. There are two lists that an AFSC is check against. The "AFSC_list" associates academic degrees and specific AFSCs (intelligence AFSC can accept any degree); and "highqual" list which identifies which academic degree are highly preferred by the Air Force for an AFSC (having a language or foreign area studies degree makes a cadet highly qualified to be in the intelligence AFSC).

Eliminating AFSC's a cadet is not qualified to do:

```
DATA A2;
SET A1;
IF AFSC NE 'NONE' THEN DO;
IF INDEX(AFSC_LIST,STRIP(AFSC)) > 0 OR
INDEX(HIGHQUAL,STRIP(AFSC)) > 0;
END;
```

When we have the final list of cadets and the AFSCs they qualify for we can start building our sparse data. We will start with defining our decision variables.

Defining the decision variables:
As mentioned before, the decision variables are the cadet/AFSC combinations. This code takes a cadet's identification number and matches it with the AFSCs the cadet qualifies for.

```
DATA ASSIGN_VAR;
LENGTH _COL_ $35;
SET A5;
COUNT = 1;
_COL_ = 'DECISION_VAR('||STRIP(ID)||','||STRIP(AFSC)||')';
RUN;
```

Once we have our decision variables, we can build our objective function and our contraints.

The Objective Function:
This is where the objective function is built and where the values for the weights for each decision variable is determined.

```
DATA OBJ (KEEP = _TYPE_ _ROW_ _COL_ _COEF_ ) SET ASSIGN_VAR;
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
/*_TYPE_*/
IF _N_ = 1 THEN _TYPE_ = 'MAX';
ELSE _TYPE_ = '.';
/*_ROW_*/
_ROW_ = 'OBJ';
/*_COEF_*/
/*Not actual formulas*/
X =  . . .;
Y =  . . .;
Z =  . . .;
_COEF_ = SUM(X, Y, Z,) /*the sum of the values assigned to the cadet's
choices, Air Force's needs and other factors. This is not the actual
formula*/
RUN;
```

The Constraints:
This piece of code ensures that all the values for cadet/AFSC combinations are integers. (We cannot assign half of a cadet to an AFSC.)

```
/*INTEGER CONSTRAINTS FOR PEOPLE*/
PROC SORT DATA = ASSIGN_VAR;
BY ID;
RUN;

DATA INTEGER (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
SET ASSIGN_VAR;
BY ID;
IF FIRST.ID THEN _TYPE_ = 'INTEGER';
ELSE _TYPE_  = '.';
_ROW_ = 'INTEGER_FLAG_'||STRIP(ID);
_COEF_ = 1;

RUN;

/*UPPER BOUNDS FOR PEOPLE*/
DATA UPPERS (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
SET ASSIGN_VAR;
BY ID;
IF FIRST.ID THEN _TYPE_ = 'UPPERBD';
```

```
ELSE _TYPE_  = '.';
_ROW_ = 'UPPERS'||STRIP(ID);
_COEF_ = 1;
RUN;
DATA INTEGER_CONSTRAINTS;
SET INTEGER UPPERS;
RUN;

This piece of code ensures that a cadet is only assigned to one AFSC.
/*PERSON CONSTRAINTS*/
PROC SORT DATA = ASSIGN_VAR;
BY ID;
RUN;

DATA MATCH_AFSC_CONSTRAINTS (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
SET ASSIGN_VAR;
BY ID;
IF FIRST.ID THEN _TYPE_ = 'EQ';
ELSE _TYPE_  = '.';
_ROW_ = 'MATCH_AFSC_'||STRIP(ID);
_COEF_ = 1;
OUTPUT;
IF LAST.ID THEN DO;
_TYPE_ = '.';
_COL_ = '_RHS_';
_COEF_ = 1;
OUTPUT;
END;
RUN;

This piece of code tells Proc LP that a certain AFSC must have at least X
number cadets assigned to it.
/*AFSC SPACES AVAILABLE TO ASSIGN*/
PROC SORT DATA = ASSIGN_VAR;
BY AFSC ID;
RUN;

DATA AFSC_LOWERBD (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
SET ASSIGN_VAR;
BY AFSC;
IF FIRST.AFSC THEN _TYPE_ = 'GE';
ELSE _TYPE_ = '.';
_ROW_ = 'AFSC_LOWERBD'||(AFSC);
_COEF_= 1;
OUTPUT;
IF LAST.AFSC THEN DO;
_TYPE_ = '.';
_COL_  = '_RHS_';
_COEF_ = LOWER;
OUTPUT;
END;
RUN;
```

This piece of code tells Proc LP that a certain AFSC must have no more than X number cadets assigned to it.

```
/*AFSC SPACES AVAILABLE TO ASSIGN*/
DATA AFSC_UPPERBD (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
LENGTH _TYPE_ $15  _COL_ $35 _COEF_ 8 _ROW_ $30;
SET ASSIGN_VAR;
BY AFSC;
IF FIRST.AFSC THEN _TYPE_ = 'LE';
ELSE _TYPE_ = '.';
_ROW_ = 'AFSC_UPPERBD'||(AFSC);
_COEF_= 1;
OUTPUT;
IF LAST.AFSC THEN DO;
_TYPE_ = '.';
_COL_  = '_RHS_';
_COEF_ = UPPER;
OUTPUT;
END;
RUN;
```

What the SAS Dataset looks like:

type	_col_	_row_	_coef_
max	(000000001,13MX)	obj	-314.9
.	(000000001,13SX)	obj	0.052
.	(000000001,14NX)	obj	0.315
.	(000000001,21AX)	obj	-314.9
.	(000000001,21MX)	obj	-314.9
.	(000000001,21RX)	obj	-314.9
.	(000000001,31PX)		
.	(000000001,33SX)		
.	(000000001,34MX)		
.	(000000001,35PX)		
.	(000000001,37FX)		
.	(000000001,61SXA)		
.	(000000001,62EXA)		
.	(000000001,62EXB)		
.	(000000001,62EXG)		
.	(000000001,63AX)		
.	(000000002,13MX)		
.	(000000002,13SX)		
.	(000000002,14NX)		
.	(000000002,21AX)		
.	(000000002,21MX)		
.	(000000002,21RX)		
.	(000000002,31PX)		

row	_type_	_col_
lowerbd_31PX	.	(000000001, 31PX)
lowerbd_31PX	.	(000000002, 31PX)
lowerbd_31PX	.	(000000003, 31PX)
lowerbd_31PX	.	(000000004, 31PX)
lowerbd_31PX	.	(000000005, 31PX)
lowerbd_31PX	.	(000000006, 31PX)
lowerbd_31PX	.	(000000007, 31PX)
lowerbd_31PX	.	(000000008, 31PX)
lowerbd_31PX	.	(000000009, 31PX)
lowerbd_31PX	.	(000000010, 31PX)
lowerbd_31PX	.	(000000011, 31PX)
lowerbd_31PX	.	(000000012, 31PX)
lowerbd_31PX	.	(000000013, 31PX)
lowerbd_31PX	.	(000000014, 31PX)
lowerbd_31PX	.	(000000015, 31PX)
lowerbd_31PX	.	(000000016, 31PX)
lowerbd_31PX	.	(000000017, 31PX)
lowerbd_31PX	.	(000000018, 31PX)
lowerbd_31PX	.	(000000019, 31PX)
lowerbd_31PX	.	(000000020, 31PX)
lowerbd_31PX	.	(000000021, 31PX)
lowerbd_31PX	.	(000000022, 31PX)
lowerbd_31PX	.	(000000023, 31PX)
lowerbd_31PX	.	under_1_31PX
lowerbd_31PX	.	under_2_31PX
lowerbd_31PX	.	under_3_31PX
lowerbd_31PX	.	_rhs_

row	_type_	_col_	_coef_
asgn_000000001	eq	(000000001,13MX)	1
asgn_000000001	.	(000000001,13SX)	1
asgn_000000001	.	(000000001,14NX)	1
asgn_000000001	.	(000000001,21AX)	1
asgn_000000001	.	(000000001,21MX)	1
asgn_000000001	.	(000000001,21RX)	1
asgn_000000001	.	(000000001,31PX)	1
asgn_000000001	.	(000000001,33SX)	1
asgn_000000001	.	(000000001,34MX)	1
asgn_000000001	.	(000000001,35PX)	1
asgn_000000001	.	(000000001,37FX)	1
asgn_000000001	.	(000000001,61SXA)	1
asgn_000000001	.	(000000001,63AX)	1
asgn_000000001	.	_rhs_	1
asgn_000000002	eq	(000000002,13MX)	1
asgn_000000002	.	(000000002,13SX)	1
asgn_000000002	.	(000000002,14NX)	1
asgn_000000002	.	(000000002,21AX)	1
asgn_000000002	.	(000000002,21MX)	1
asgn_000000002	.	(000000002,21RX)	1
asgn_000000002	.	(000000002,31PX)	1
asgn_000000002	.	(000000002,34MX)	1
asgn_000000002	.	(000000002,35PX)	1
asgn_000000002	.	(000000002,37FX)	1
asgn_000000002	.	(000000002,65FX)	1
asgn_000000002	.	(000000002,65WX)	1
asgn_000000002	.	_rhs_	1

Creating the Sparse Data:

These lines of code create the sparse data for the Proc LP procedure. The sparse data is the collection of all datasets from the objective function and the constraints.

```
DATA SPARSE (KEEP = _TYPE_ _ROW_ _COL_ _COEF_);
SET OBJ
    INTEGER_CONSTRAINTS
    MATCH_AFSC_CONSTRAINTS
    AFSC_LOWERBD
    AFSC_UPPERBD;
RUN;
```

Solving the Optimization Problem:
As with the example, these lines of SAS code below will solve our problem and will provide us with an optimized solution. It will also provide problem summary, solution summary, variable summary, and constraint summary to use in an analysis of the problem.

```
PROC LP SPARSEDATA;
   TYPE _TYPE_;
   ROW _ROW_;
   COL _COL_;
   COEF _COEF_;
ODS OUTPUT VARIABLESUMMARY = RESULTS;
RUN;
```

CONCLUSION

This presentation has shown us how SAS OR's Proc LP helps in the process of assigning ROTC cadets to an AFSC. It describes the elements of an LP problem and how these elements are translated into SAS language. We provided a simple example that mimics the ROTC Classification problem to better understand the transition from a mathematical formulation to a SAS formulation. We discussed the format in which the Proc LP sparse data needs in order for SAS OR's Proc LP procedure to calculate the optimum solution for a problem. SAS's Proc SQL procedure was introduced to help build the sparse data, rather than entering the data by hand. Overall, SAS's OR Proc LP procedure and SAS's Proc SQL procedure has made the process of assigning ROTC cadets to an AFSC easier and more consistent. Rather than looking at a list of cadets and a list of available of AFSCs and matching them manually, we now have a model that assigns cadets to an AFSC in a calculated and fair process for the cadets, while still meeting AF requirements.

Contact Information

Anthony Lawrence
Kristen Cavallaro
United States Air Force, Air Force Personnel Center, DPAPA
550 C St West Ste 36
Randolph AFB, TX 78150-4701
Work Phone: 210-565-4261
Fax: 210-565-3503
E-mail: Anthony.Lawrence.1@us.af.mil / Kristen.Cavallaro@us.af.mil